

Department of Veterans Affairs

Compensation and Pension Records Interchange (CAPRI)

System Design Document Patch 187



**December 2014
Version 1.9**

Revision History

Date	Version	Description	Author
12/03/2014	1.9	Update all sections removing template instruction text and make final edits.	
12/01/2014	1.9	Update CodeCR570 send vendor exam results to VLER DAS from in-house.	
11/26/2014	1.8	Update with CodeCR568 and CodeCR570	
9/30/2014	1.7	Update with system design info from Mark Battler	
5/27/2014	1.6	Update with CodeCR562 and CodeCR565	
5/27/2014	1.5	Update with CodeCR563	
5/27/2014	1.4	Update with CodeCR567 and CodeCR577	
3/17/2014	1.3	Update Approval Signature Section with AERB 'Approved' certificate.	
2-24-2014	1.2	Updated section 1.5 based on AERB review	
2-13-2014	1.1	Added proxy server details	
9-11-2013	1.0	Initial documentation effort for Patch 187 Milestone 1.	

Contents

1. Introduction	1
1.1. Purpose of this document.....	1
1.2. Identification	1
1.3. Scope	1
1.4. Relationship to Other Plans.....	1
1.5. Methodology, Tools, and Techniques.....	2
1.6. Constraining Policies, Directives and Procedures	2
1.7. Constraints	2
1.8. Design Trade-offs	3
1.9. User Characteristics	3
1.10. User Problem Statement	3
2. Background	4
2.1. Overview of the System	4
2.2. Overview of the Business Process	5
2.3. Assumptions	5
2.4. Legacy System Retirement.....	5
3. Conceptual Design.....	6
3.1. Conceptual Application Design	6
3.1.1. Application Context.....	6
3.1.2. High-Level Application Design	12
3.1.3. High-Level Application Design Enhancements.....	12
3.1.4. Application Locations	12
3.1.5. Application Users	12
3.2. Conceptual Data Design.....	13
3.2.1. Project Conceptual Data Model	13
3.2.2. Database Information	13
3.2.3. User Interface Data Mapping	13
3.3. Conceptual Infrastructure Design	13
3.3.1. System Criticality and High Availability.....	13
3.3.2. Special Technology	13
3.3.3. Technology Locations.....	13
3.3.4. Conceptual Infrastructure Diagram.....	13
4. System Architecture	16
4.1. Hardware Architecture	16
4.2. Software Architecture.....	18
4.3. Network Architecture.....	19

4.4. Service Oriented Architecture / ESS	19
4.5. Enterprise Architecture	20
5. Data Design	22
5.1. DBMS Files	22
5.2. Non-DBMS Files	22
5.3. Data View	22
6. Detailed Design	23
6.1. Hardware Detailed Design.....	23
6.2. Software Detailed Design.....	23
6.2.1. Conceptual Design	23
6.2.2. Specific Requirements	24
6.3. Network Detailed Design.....	174
6.4. Service Oriented Architecture / ESS Detailed Design	174
6.4.1. Service Description for <Consumed Service Name>.....	174
6.4.2. Service Design for <Provided Service Name>	174
7. External System Interface Design.....	175
7.1. Interface Architecture.....	175
7.2. Interface Detailed Design	186
Notes Tab	187
C&P Exam Requests.....	193
Patient Profile Medical.....	193
administration Service (MAS) (Full).....	193
8. Human-Machine Interface	199
8.1. Interface Design Rules	199
8.2. Inputs	199
Logging onto CAPRI.....	200
VistA Terminal.....	200
8.3. Outputs	206
C&P Exam Requests.....	207
8.3.1. View/Edit Selected Request.....	207
7131 Request.....	210
8.3.2. Add a New 7131 Request	210
8.3.3. 7131 Request Status Inquiry	214
CAPRI Reports (Patient-Specific)	215
8.3.4. Patient Inquiry.....	215
8.3.5. Detailed Inpatient Inquiry	215
8.3.6. C&P Exam Detail	215
8.3.7. 7131 Detail.....	215

8.3.8.	Additional Treating Facilities	216
8.3.9.	View Registration Data	216
8.3.10.	Patient Profile Medical Administration Service (MAS) (Full).....	216
8.3.11.	Surgery Report.....	216
8.3.12.	Other Patient-Specific Reports	216
8.3.13.	Reprint a 21-Day Certificate	216
8.3.14.	Reprint a Notice of Discharge.....	216
	CAPRI Reports (Non-Patient-Specific)	216
8.3.15.	C&P Exams Reports	217
8.4.	Navigation Hierarchy	223
	Provide a diagram of the navigation hierarchy that shows how a user moves through the GUI.....	223
8.4.1.	Start- Capri Select a Patient.....	224
8.4.2.	Start-Capri Create a New Patient.....	225
8.4.3.	Create a Compensation and Pension Request	226
8.4.4.	Create a New 7131 Request	227
8.4.5.	Reports	228
8.4.6.	Cancellations	229
9.	Security and Privacy.....	234
9.1.	Security.....	234
9.1.1.	Security Control Families.....	234
9.2.	Security Management.....	236
9.3.	General Security	236
9.3.1.	Remote Systems	236
9.3.2.	Contingency Planning	237
9.3.3.	Interfacing	237
9.3.4.	Electronic Signatures	237
9.3.5.	Security Keys	237
9.4.	Privacy	237
9.4.1.	Privacy is maintained through system access restrictions based on VA DIRECTIVE 6500 InformationSecurity	237
9.5.	File Security	237
10.	Approval Signatures	238

1. Introduction

The Compensation and Pension Record Interchange (CAPRI) project is an information technology initiative to improve service to disabled veterans by promoting efficient communication between the Veterans Health Administration (VHA) and Veterans Benefits Administration (VBA). Online access to medical data enhances the timeliness of the benefits determination. Previous attempts to automate this process were hindered by the "roll and scroll" nature of the VHA computer interface of the Automated Medical Information Exchange (AMIE) II. The CAPRI software acts as a bridge between the VBA and VHA information systems. It offers VBA Rating Veteran Service Representatives and Decision Review Officers help in building the rating decision documentation through online access to medical data. It also offers VHA Compensation and Pension (C&P) staff an easy, standardized way of recording C&P Examination reports.

Initially developed specifically for VBA, the utility of CAPRI has been expanded to other user groups that include VHA, Office of the Medical Inspector, Office of Information (OI), Research, and Veteran Service Officers. Recently, most of the newest features of CAPRI are specifically targeted at adding features to be used by VHA C&P providers and staff.

1.1. Purpose of this document

The purpose of this document is to provide an overview of the current state of the system design. This document provides a high-level overview of the Compensation and Pension Record Interchange application. This document translates the VA requirements identified in the Performance Work Statement (PWS) into a document from which the CAPRI development team can create the actual system. It identifies the top-level system architecture, and identifies hardware, software, communication, and interface components.

1.2. Identification

The CAPRI Contract is Contract number is VA118-11-F-0480.

1.3. Scope

The scope of Phase II VLER DAS is for CAPRI to have the ability to retrieve DBQs from the now official DBQ central data store, VLER DAS SOR. Modifications to the CCR functionality will also occur as Vendors start to transmit their DBQs to VLER DAS. CAPRI will need to be able to retrieve the vendor DBQs from VLER DAS instead of the current SFTP method. Since vendors will start transmitting to VLER DAS at different times, CAPRI will need to be able to support both the SFTP file grab and VLER DAS query.

1.4. Relationship to Other Plans

The following CAPRI documents are related to this document:

- CAPRI User Manual (UM)
- CAPRI Release Notes (RN)
- CAPRI System Administration and Technical Guide
- CAPRI Software Design Document (SDD)
- CAPRI Specification Documents (RSD)

- CAPRI Requirements Traceability Matrix (RTM)
- CAPRI Project Management Plan (PMP)
- CAPRI Master Test Plan (MTP)
- CAPRI Production Operations Manual (POM)

1.5. Methodology, Tools, and Techniques

Table 1-1: CAPRI Methodology and Tools List

Tool	Design task
Rational ClearQuest	Track development of Change Requests and Code Change Requests from approval to deployment.
Rational ClearCase	Provide version control for database scripts, data modeling reports, and documentation.
Delphi 2006	Support CAPRI legacy application
Delphi XE2	Support the Authentication application on the proxy server used to authenticate a CAPRI user prior sending transmissions to VLER/DAS. Support web service transmission to Virtual VA.
Windows Enterprise 2008 R2	Virtual Machine(s) hosting the CAPRI Proxy Servers.
Java Version 1.6.0_71	Proxy Server application controlling traffic between CAPRI, the Authentication Server, and VLER/DAS.
WebLogic 12c	Java EE application server that manages and executes the CAPRI Proxy service.
InterSystems Cache' version 2011.1.2.7010.12942	Supports CAPRI data retrieval from Vista systems.

1.6. Constraining Policies, Directives and Procedures

The System Design Document follows at a minimum the constraints and requirements of the ProPath template for this document. This ProPath template has been expanded to include additional content, specifically more detailed system and hardware architecture sections. This document must conform to the 508 policy for VA.

1.7. Constraints

Due to the short turnaround time for producing this document, not all areas have information available.

1.8. Design Trade-offs

CAPRI architecture was designed with user requirements consideration for Flexibility, Interoperability, Performance, and Reliability to deliver optimum performance.

1.9. User Characteristics

CAPRI's user community is primarily comprised of Veterans Benefits Administration (VBA) and Veterans Health Administration (VHA) personnel. There are also Department of Defense (DoD), VA Office of Inspector General, Veteran Service Organizations (VSO) and National Cemetery Administration users.

1.10. User Problem Statement

N/A

2. Background

CAPRI provides VBA employees with a standardized, user-friendly method to rapidly access veterans' electronic medical records throughout the Department of Veterans Affairs (VA). CAPRI delivers leading edge "point and click" technology to the users' desktops. In addition, the learning curve for CAPRI is significantly less than that for character-based systems. CAPRI builds upon existing VHA information security approaches. In addition to using established mechanisms to ensure only authorized access to medical data, CAPRI adds a level of security by allowing VBA users to read but not alter electronic medical record information. CAPRI also provides innovative improvements for medical centers by integrating highly detailed (C&P) Rating examination results into the veterans' medical records. Previously, these reports were not retained online in medical center computer systems but were archived onto paper. This procedure precluded the sharing of clinically useful data.

2.1. Overview of the System

In support of the Veterans Lifetime Electronic Record (VLER) Data Access Service (DAS) and Veterans Benefit Management System (VBMS) efforts to receive and process computable data, changes will be necessary to CAPRI, PNCS, and the individual CAPRI DBQ templates to transmit a XML file of DBQ data elements through the VLER DAS.

In response to changes in technical dependencies:

1. VBMS requires ability for a PDF to be transmitted with the XMLfile.
2. The Harris solution has been put on hold; duration unknown. We will no longer use the XML schema provided by Harris.
3. New terminology standards will need to be established. Scope of terminology can add complexity.
4. VA will design XML schema to fit the requirements.
5. VLER DAS web service requires two-way SSL (also known as mutual authentication). CAPRI has been modified to relay VLER DAS web service requests through the CAPRI Proxy service using one-way SSL.
6. Due to the fact that VA will be integrating the terminology standards for the XML and designing the XML schema, code changes in PNCS will be required to enable DBQ data to be read and analyzed by other VA systems.

2.2. Overview of the Business Process

Figure 4.1: Business Processes Diagram

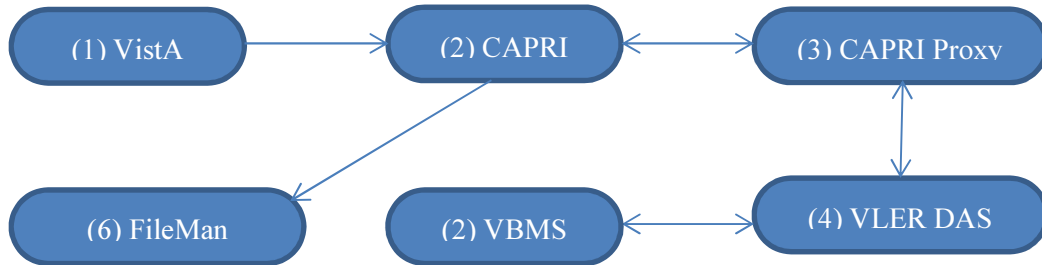


Table 2-1: Business Process

Business Process ID	Business Process Name	Type	Owner	Description
1	VistA	Existing	VistA	VistA is linked to CAPRI as the backbone of the system and CAPRI is the front end application
2	CAPRI	Modified	VistA/Fileman	The first functionality is that a C&P exam is created in CAPRI for a patient. Second, the DBQ associated to the exam is generated and completed.
3	CAPRI Proxy	Existing	CAPRI	Web service used by the CAPRI to forward a DBQ to VLER DAS using two-way SSL.
4	VLER DAS	Modified	CAPRI	The logic of CAPRI that transmits files to VLER DAS
5	VBMS	Existing	VBMS	XML and PDF file of exam results is sent to VBMS where it can be read as an item associated to the patient.
6	Fileman	Modified	VistA	XML is now transmitted to the 2507 Exam file (396.4) and the exam stores the data in XML format. This data is associated to the exam.

2.3. Assumptions

N/A

2.4. Legacy System Retirement

N/A

3. Conceptual Design

3.1. Conceptual Application Design

3.1.1. Application Context

CAPRI application interacts with other systems as described in the diagram **Error! Reference source not found..** CAPRI utilizes multiple interfaces from external applications/systems. These include VistA RPC, WebServices, Secure FTP, Shell execute calls for Windows and Interprocess communication through CCOW RPC Broker. Currently Virtual VA and VLER/DAS are connected using Webservices to CAPRI GUI client application. CAPRI VistA application connects to MVI using HealtheVet Webservice VistA client on the VistA server side. Third party vendors are communicated exam request and the exam results retrieved via Secure FTP. Single context sign-on communication is provided between CAPRI and CPRS as well as any other Windows Client application utilizing RPC Broker on the client workstation. Single context sign-on messaging is also utilized between one authenticating VistA/CAPRI server and other VistA/CAPRI servers within VA's network via a triangulating multi-part token messaging mechanism using RPC Broker's Broker Security Enhancement features. CAPRI also utilizes third party applications Adobe Acrobat and Microsoft Office Document Imaging as external document viewers and uses OLE automation to control Microsoft Word as a document editor.

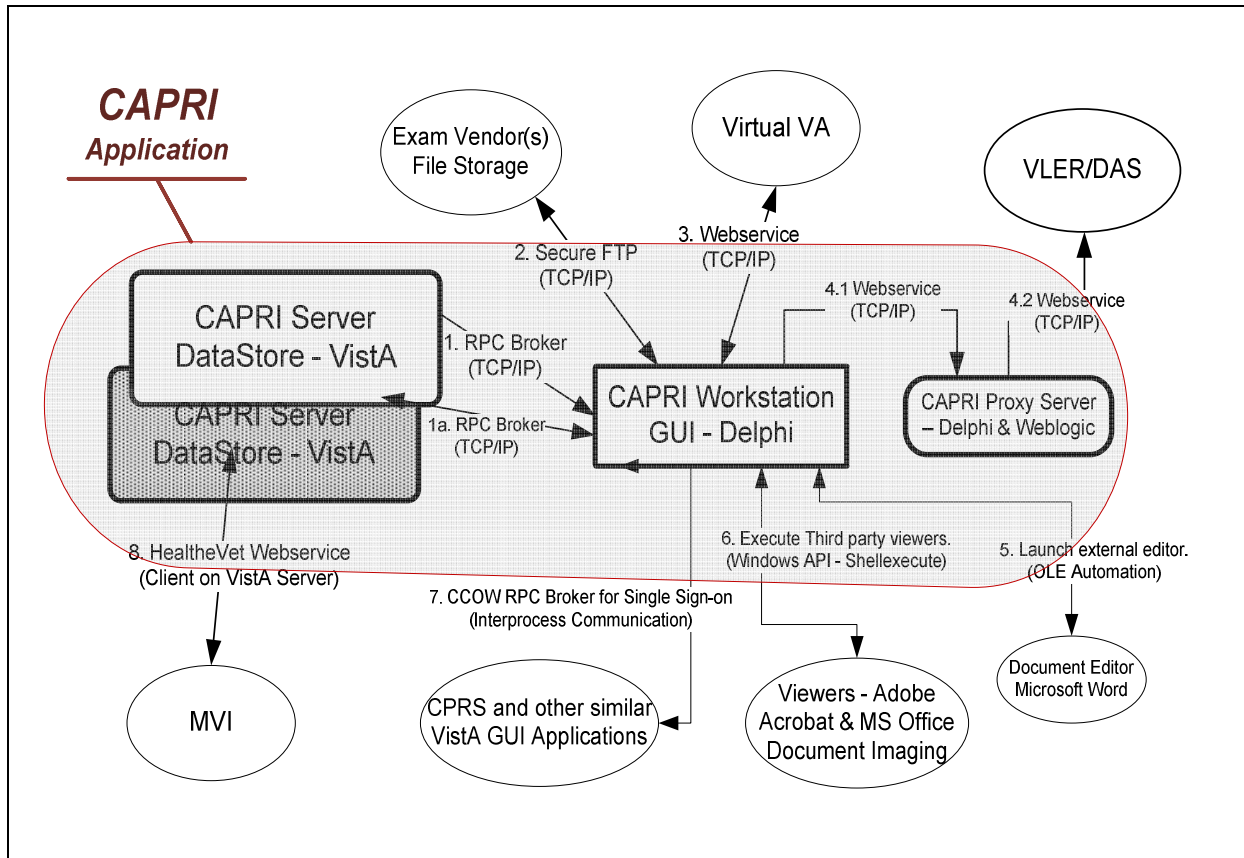


Figure 3-1: Application Context Diagram

Table 3-1: Application Context Description**Object**

ID	Name	Description	Interface Name	Interface System
1	VistA database connection	Primary communication channel between CAPRI GUI and VistA server	RPC Broker	CAPRI GUI and Server interconnect
2	Third party Exam Vendors	Contracted Vendors processing Exam Requests sent by Providers and providing results for those requests.	SFTP	CAPRI
3	Virtual VA	Storage and retrieval of patient electronic documents	Webservice	CAPRI
4	VLER/DAS	Storage and retrieval of exam results for both internal to the VA exams and external vendors.	Webservice	CAPRI
5	External Document Editor	Third party application MS-Word	OLE Automation	CAPRI
6	External Document Viewers	Third party viewers for PDF and TIFF files using Adobe Acrobat and MS Office Document Imaging respectively.	Shellexecute	CAPRI
7	CCOW RPC Broker	Single sign-on capability to allow users to connect with VistA servers on one CCOW enabled application and then users can omit having to supply credentials again for any other CCOW enabled application	Interprocess communication	CAPRI, CPRS and others
8	HealtheVet	Perform MVI Search from a VistA server using HealtheVet webservice.	Webservice	CAPRI

Interfaces External to OIT

ID	Name	Related Object	Input Messages	Output Messages	External Party
2	Third party Exam Vendors	Contracted Vendors processing Exam Requests sent by Providers and providing results for those requests.	VA uploads (sends) to the third party vendor's SFTP server an XML file containing a Request comprising of one or more exams to be performed for a patient. The request is generated by a provider using CAPRI GUI. CAPRI also sends a "Marked Insufficient" XML file to vendors for exams where the results provided have to be re-processed by the vendor.	VA receives text files as results of Exam Request sent from the third party vendor. The download (receiving) file is in plain text format. It is created by the vendor on their SFTP server and is stored in the patient record automatically by CAPRI GUI.	Third party contracted Exam Vendors

Interfaces Internal to OIT

ID	Name	Related Object	Input Messages	Output Messages	External Party
1	RPC Broker	Primary communication channel between CAPRI GUI and backend VistA server data storage.	CAPRI GUI accepts user input for multiple business functions it supports and translates it into appropriate VistA FileMan commands and data streams to transmits those to a backend VistA server. CAPRI also processes VistA server responses and appropriately presents those to the user via the GUI interface.	VistA server interprets all communication from CAPRI GUI and appropriately handles the requests. Requests include but aren't limited to User authentication, patient related information search, add, update and delete (CRUD).	N/A

ID	Name	Related Object	Input Messages	Output Messages	External Party
1a	RPC Broker	Special communication channel specifically for multi-server authentication requests	CAPRI GUI requests security tokens from an authenticating server and presents those to a new server for which “authentication” is required. Handles the resulting response from the “authentication” server.	VistA server processes security tokens and authenticating server information and makes a backend connection to validate the token and authentication information provided. Provides a response to the GUI.	N/A
3	Webservice	Virtual VA	CAPRI sends patient electronic reports for Storage in Virtual VA. CAPRI also may request a list of documents available per patient and request retrieval of patient’s documents.	Virtual VA provides confirmation of document storage. VVA also provides documents in response to retrieval requests.	Virtual VA
4	Webservice	VLER/DAS	CAPRI sends patient exam results using XML to VLER/DAS. CAPRI may also request a list of exam results. A single exam may be viewed and reviewed for signature.	VLER/DAS provides confirmation of document storage. VLER/DAS provides documents in response to retrieval requests.	VLER/DAS -> VBMS
5	External Editor	MS-Word	CAPRI creates a draft Word Document file with pre-filled information and launches MS-word to allow user to edit and save this document. The document is		N/A
6	Document Viewers	Adobe Acrobat Microsoft Office Document Imaging	Third party apps are launched as viewers for documents (retrieved from Virtual VA)	N/A	N/A

ID	Name	Related Object	Input Messages	Output Messages	External Party
7	CCOW RPC Broker	CPRS and other applications	Single sign-on functionality	Authentication information is exchanged between applications through interprocess communication	N/A
8	HealtheVet webservice	MVI Search webservice	Patient query criteria	Zero or more patient query results	MVI

Externally Shared Data Stores

ID	Name	Data Stored	Owner	Access
N/A	N/A	N/A	N/A	N/A

3.1.2. High-Level Application Design

N/A

3.1.3. High-Level Application Design Enhancements

The CAPRI application will be enhanced to enable the following:

- CAPRI will have the ability to retrieve and display Vendor exam results (XML files) that were transmitted directly to VLER DAS.
- CAPRI will have the ability to review exam results
- CAPRI will have the ability to provide approval or rejection of exam results and transmit/communicate back to VLER DAS
- Create the ability to communicate an approval or rejection status to the vendor through CCR, including comments to note issues requiring correction
- Create the ability to store the exam result back into the 2507 exam request if one exists for the Veteran; thus allowing the facility to check the referred out exam back in using CCR functionality
- When the examiner is signing/completing multiple templates merged together, create the ability to break the current merged/concatenated exam report back into single exams for purpose of PDF file creation
- CAPRI will have the ability to retrieve alerts from VLER DAS for “PendingUnderReview” exams.

3.1.4. Application Locations

N/A

3.1.5. Application Users

Table 3-2: Application Users

Application Component	Location	User
C & P Exam Requests	VA sites nationwide	VBA C & P Examiners
DBQ Processing	VHA facilities	VHA Personnel
Reports	VBA Regional Offices and VA Medical Centers	VBA and C&P Staff
VistA button	Remote Users (Regional Offices or VA Medical Centers)	HRC Staff
Vocational Rehab Tab	VBA Regional Offices and VA Medical Centers	VBA and VHA Vocational Rehab staff
CAPRI Contract Referral	VA Medical Centers	C&P Staff
CAPRI Remote Menus	Nationwide VBA, HIA	VBA and HIA Staff

3.2. Conceptual Data Design

3.2.1. Project Conceptual Data Model

The CAPRI application utilizes the underlying data model of the VistA system. For details of the VistA system please refer to relevant documents on VDL website.

3.2.2. Database Information

N/A

3.2.3. User Interface Data Mapping

N/A

3.2.3.1. Application Screen Interface

N/A

3.2.3.2. Application Report Interface

N/A

3.3. Conceptual Infrastructure Design

The CAPRI application utilizes the underlying Infrastructure of the VistA system. For details of the VistA system please refer to relevant documents on VDL website.

3.3.1. System Criticality and High Availability

N/A

3.3.2. Special Technology

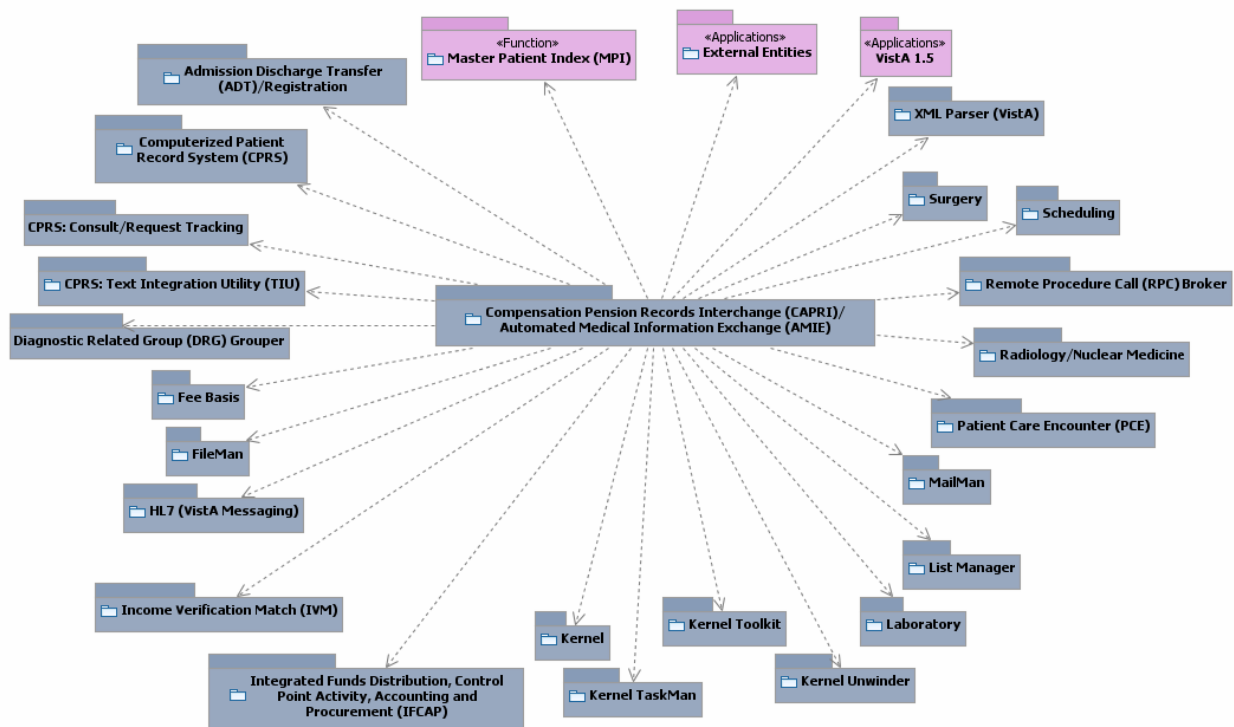
The CAPRI proxy service must be enhanced to support the workflow for:

1. C&P examination to vendors
2. “PendingUnderReview” event notification

3.3.3. Technology Locations

N/A

3.3.4. Conceptual Infrastructure Diagram



3.3.4.1. CAPRI Software Services

- Patient Selection
- C&P Exam Requests
- 7131 Request
- Reports (Patient Specific, Non Patient Specific, Consolidated Remote Reports)
- Health Summaries
- Compensation and Pension Worksheet Module (CPWM)
- Vocational Rehab
- CAPRI Remote Functionality
- Vista Terminal
- Virtual VA - Transmit and Retrieve Documents
- VLER SOR – Transmit and Retrieve C&P Exams
- Validate the Patient Restricted List
- CAPRI Contract Referral (CCR)
- DOD Records
- VistAWeb

3.3.4.2. CAPRI Connected Systems

Note: Please see diagram and descriptions below from CAPRI's System Design Document.

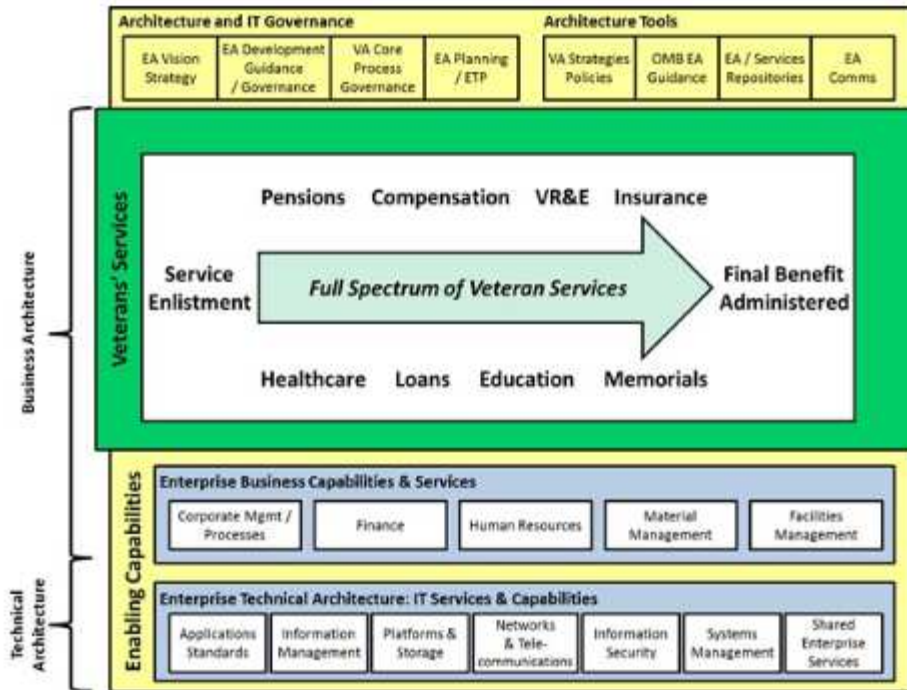
- VistA Servers – VAMC Hospitals
- Claims Server – CLAIMS (Martinsburg, WV)
- VistA / CPRS – VAMC Hospitals

- Station 200 and BHIE
- Master Veteran Index
- FHIE / BHIE Server
- CAPRI Contract Referral (CCR) – communicate via SFTP and VLER/DAS to Contracted Vendors

4. System Architecture

This section describes the system architecture of the CAPRI product. It includes detailed information about the technical architecture and components associated with CAPRI

Capri is a part of the [OneVA EA](#) Content Scope. Illustrated in the diagram below from that site.

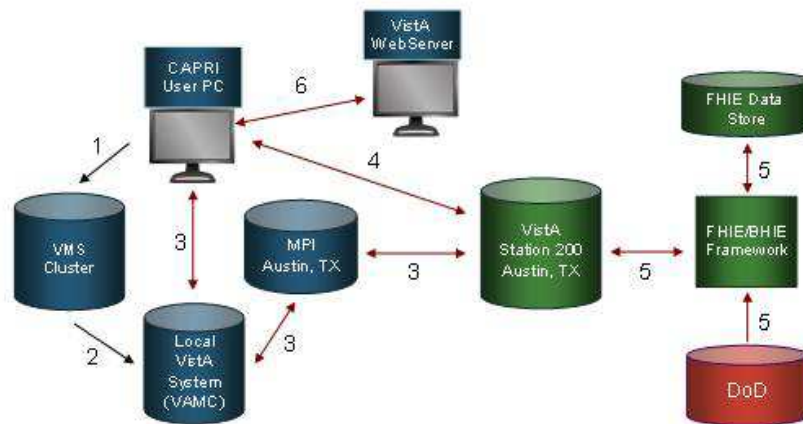


4.1. Hardware Architecture

The diagram associated with the following link is a high-level association of CAPRI and the systems it interfaces to. This diagram, "CAPRI Data Flow Diagram_Updated09May2012" is available at the following URL:

[Redacted URL]

CAPRI Data Flow Diagram



1. Authorized remote CAPRI User (usually VBA) logs into the VMS Cluster (if VHA local user - skip to step 3)
2. The VMS Cluster validates the list of VHA facilities the user is authorized to connect to and passes the CAPRI session over to the selected VistA system (CAPRI connection to the VMS Cluster is dropped and user is now connected to that VistA system)
3. CAPRI user selects patient from the VistA system; that process initiates the retrieval of the list (through the MPI) of other facilities where that the patient has presented for care; that list will include Station 200 if DoD data is available for the patient
4. If DoD data is available on Station 200, a separate DoD tab is enabled in CAPRI; when DoD tab is selected, CAPRI queries Station 200 for the available DoD report types and presents to the user
5. User picks the desired DoD report type and date range, then CAPRI queries Station 200; Station 200 queries the BHIE framework for the needed DoD data elements for the selected report type/date range (limited data is held at VA)
6. Using authentication established when user connects to CAPRI, the user is connected to VistA.

CAPRI's hardware infrastructure is centralized and located in Austin, TX. It consists of a VistA Station 200, MPI, CAPRI User PC (Thousands of VA Workstations nationwide), Two (2) CAPRI Authentication/Proxy Server and depends on the Local VistA System (VAMC), VistA Web Server, FHIE Data Store, and the FHIE/BHIE Framework as described in the following table:

Hardware	Qty	Centralized (y/n)	Location Support (OIFO)	Type (workstation /server)	OS (Operating System)
VistA Station 200			Austin, TX		
MPI			Austin, TX		
CAPRI User PCs (VA Workstations)	10,000 +	No	Nationwide	Workstation	Windows 7

CAPRI Authentication/Proxy Server	2	Yes	Austin, TX	Cluster of Servers	Windows Server 2008
Additional Hardware Dependencies from VistA					
Local VistA System (VAMC)			Nationwide		
VistA Web Server			Nationwide		
FHIE Data Store			Nationwide		
FHIE/BHIE Framework			Nationwide		

System: OS OpenVMS V8.3

, hp AlphaServer ES80 7/1150
, hp AlphaServer ES80 7/1150
, hp AlphaServer ES47 7/1150
, hp AlphaServer ES47 7/1150

HP EVA8000 SANS Storage Array

HP MSL6000 Tape Library

4.2. Software Architecture

CAPRI's software is comprised of the following programs:

OS OpenVMS V8.3

Cache for OpenVMS/ALPHA V8.x (Alpha) 2011.1.2 (Build 701 + Adhoc 12942) 19-JUL-2013

Software Inventory	Qty	Location Support (OIFO)	Hosting server (Hardware)
VistA/Fileman	135+	Nationwide	Various
CAPRI GUI Suite	30,000+	Nationwide	Various
CAPRI Proxy	2	Austin, TX	Virtual Servers
CAPRI Authentication Module	2	Austin, TX	Virtual Servers

Software Inventory	Operating System	Database	Compilers	Communications Software	Other, Development Tools, Programming Languages used, etc.
VistA/Fileman	Linux and VMS	CACHE			
CAPRI GUI Suite	Windows 7		Delphi		Rational ClearCase
CAPRI Proxy	Windows 2008 Server		JAVA	Web Logic	Rational ClearCase
CAPRI Authentication Module	Windows 2008 Server		Delphi		Rational ClearCase

4.3. Network Architecture

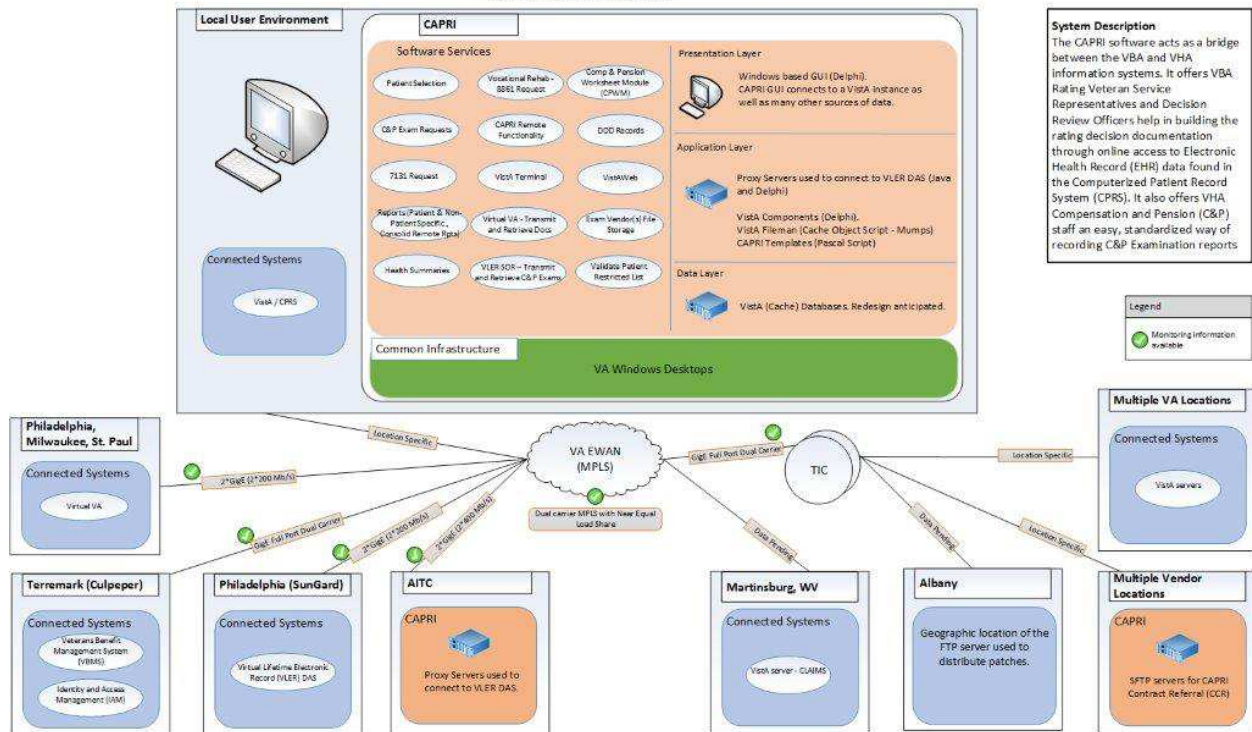
Hardware Type (i.e. NAS, Core Switch, Distribution Switches, etc).	Qty	Location Support (OIFO)	Connections (i.e. NAS to Core Switch; Distribution Switches to parsers, Wireless Access Points, etc.)
VA Intranet Infrastructure		Nationwide	

4.4. Service Oriented Architecture / ESS

1. **Service Oriented Architecture:** A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.
2. **Service workflow provided by Patch DVBA*2.7*187** accreditation

Compensation & Pension Record Interchange (CAPRI) and Connected Systems

Version: February 27th, 2014



HeathVet Web Services Client HWSC supports SOAP synchronous web service access:

Link- [REDACTED]

- SOAP (Service Oriented Architecture Protocol) – a formal XML-based protocol for accessing services

VLER The Virtual Lifetime Electronic Record (VLER) Health program gives VA and non-VA health care providers secure access to certain parts of your electronic health record. This access reduces the need for Veterans and their families to request and carry paper medical records from one health care provider to another, and provides other potential benefits to Veterans and their providers

4.5. Enterprise Architecture

The current TRM/SP is located VA Enterprise Architecture (EA) v2.1 at [REDACTED]

The OneVA EA HOME page is located at [REDACTED]

The Capri Enterprise Archectecture complies with the OneVA EA Standard

[REDACTED]

5. Data Design

This section outlines the design of the database management system (DBMS) and non-DBMS files associated with the system.

5.1. DBMS Files

Data Management can be found in the Capacity Management Tools Technical Manual

Data Management Details can be found in the Capacity Management Tools User Manual at:

This document covers topics that may include the following:

- Logical model; provide normalized table layouts, entity relationship diagrams, and other logical design information
- DBMS schemas, subschemas, records, sets, tables, storage page sizes
- Access methods (such as indexed, via set, sequential, random access, sorted pointer array)
- Estimate the database file size or volume of data within the file, data pages, including overhead resulting from access methods and free space
- Definition of the update frequency of the database tables, views, files, areas, records, and sets
- Estimates on the number of transactions that the database may have to process.
- VISTA SQL Interface information can be found at

5.2. Non-DBMS Files

Non-DBMS Files information involved including the items listed below can be found in the VISTA MONOGRAPH at

5.3. Data View

Description of the VISTA DataView can be found in the VISTA MONOGRAPH at

The following Excerpt from this document is an introduction to VISTA MONOGRAPH: “This *Monograph* provides an overview of the **V**ETERANS HEALTH INFORMATION **S**YSTEMS AND **T**ECHNOLOGY **A**RCHITECTURE information system—**Vista**—used by the Veterans Health Administration (VHA) of the U.S. Department of Veterans Affairs (VA) in serving America’s veterans through the provision of quality health care which enhances our Veterans’ health and well-being

6. Detailed Design

6.1. Hardware Detailed Design

CAPRI (VistA/AMIE) utilizes existing hardware infrastructure, VistA servers and local user Windows workstations. Hence there is no separate Hardware Detailed Design document.

CAPRI Proxy Servers are configured as follows.

- Network load balanced, F-5 (shared in the AITC data center)
- VMWare virtual machines Qty 2
- 8 GB RAM
- 40 GB shared storage
- Windows 2008 Server R2 Enterprise Edition
- Oracle Weblogic 12c
- Located

6.2. Software Detailed Design

This section focuses on changes and enhancements along with new features being introduced in CAPRI in the current patch.

CAPRI is a large application developed over many years. There are a large number of RSD's that have been created over the years. This SDD document only lists the changes introduced in this patch and references the relevant CodeCR numbers. For a full list of CAPRI System Features reference CAPRI Users Guide and other CAPRI documents as well as previous versions of CAPRI SDD (Software Design Documents).

6.2.1. Conceptual Design

Instructions for this section call for “how the software will be built”. However CAPRI System is already in operation and in the current patch there were no “Conceptual Design” changes. Hence there is no design specific information available to provide for this section.

6.2.1.1. Product Perspective

N/A

6.2.1.1.1. User Interfaces

N/A

6.2.1.1.2. Hardware Interfaces

N/A

6.2.1.1.3. Software Interfaces

N/A

6.2.1.1.4. Communications Interfaces

N/A

6.2.1.1.5. Memory Constraints

N/A

6.2.1.1.6. Special Operations

N/A

6.2.1.2. Product Features

N/A

6.2.1.3. User Characteristics

N/A

6.2.1.4. Dependencies and Constraints

N/A

6.2.2. Specific Requirements

6.2.2.1. Database Repository

N/A

6.2.2.2. System Features

CAPRI is a large application developed over many years. There a large number of RSD's that have been created over the years. This SDD document only lists the changes introduced in this patch and references the relevant CodeCR numbers. For a full list of CAPRI System Features reference CAPRI Users Guide and other CAPRI documents.

6.2.2.3. CODECR567 – VLER DAS Security Keys

6.2.2.3.1. Routines (Entry Points) – Not applicable for CodeCR 567

6.2.2.3.2. Templates – Not applicable for CodeCR 567

6.2.2.3.3. Bulletins – Not applicable for CodeCR 567

6.2.2.3.4. Data Entries Affected by the Design – Not applicable for CodeCR 567

6.2.2.3.5. Unique Record(s) - Not applicable for CodeCR 567

6.2.2.3.6. File or Global Size Changes – Not applicable for CodeCR 567

6.2.2.3.7. Mail Groups – Not applicable for CodeCR 567

6.2.2.3.8. Security Keys

Table 26: Security Keys

Security Keys	Activities
----------------------	-------------------

Security Keys	Activities
Security Key Name	DVBA CAPRI GetDocsFromVLER
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	

Related Routines	Routines “Called By”	Routines “Called”
N/A		

Security Keys	Activities
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Security Key Description	The key will allow users to run queries to VLER DAS
Subordinate Keys	N/A
Mutually Exclusive Keys	N/A
Granting Condition Logic	N/A

Current Logic
N/A

Modified Logic (Changes are in bold)
N/A

Security Keys	Activities
Hierarchical Precedence	N/A

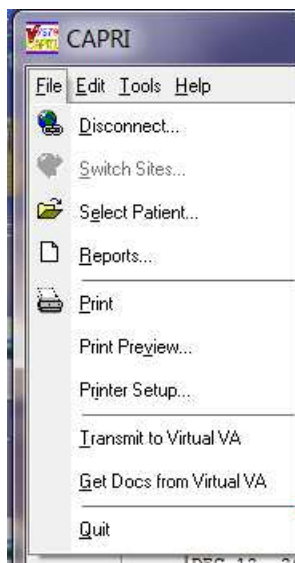
- 6.2.2.3.9. Options – Not applicable for CodeCR 567
- 6.2.2.3.10. Protocols – Not applicable for CodeCR 567
- 6.2.2.3.11. Remote Procedure Call (RPC) – Not applicable for CodeCR 567
- 6.2.2.3.12. Constants Defined in Interface – Not applicable for CodeCR 567
- 6.2.2.3.13. Variables Defined in Interface – Not applicable for CodeCR 567
- 6.2.2.3.14. Types Defined in Interface – Not applicable for CodeCR 567
- 6.2.2.3.15. GUI

Table 36: GUI

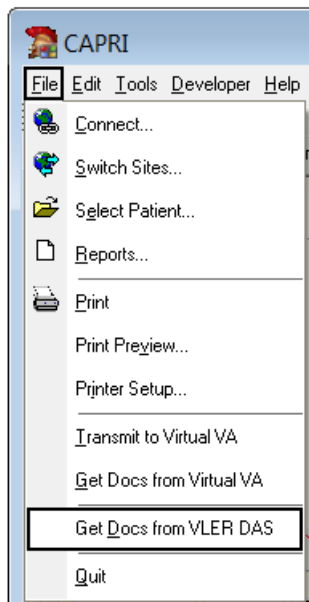
Unit Name	Description
	Main.pas

- 6.2.2.3.16. GUI Classes – Not applicable for CodeCR 567

- 6.2.2.3.17. Current Form



6.2.2.3.18. Modified Form



6.2.2.3.19. Components on Form

Table 39: Components on Form

Name	Type	Description
actFileRetrieveDocsVLERDAS	TActionClientItem	Menu Item: Get Docs from VLER DAS
actFileSeparatorBarVisible	TActionClientItem	Menu Item: Separator Bar

6.2.2.3.20. Events

Table 40: Events

Name	Type	Description
actFileRetrieveDocsVLERDASUpdate	Procedure	Routine added to support enabling and disabling of the 'Get Docs from VLER DAS' menu item
actFileConnectExecute	Procedure	Routine modified to control menu item visibility based on the security key at startup.

- 6.2.2.3.21. **Methods – Not applicable for CodeCR 567**
- 6.2.2.3.22. **Special References – Not applicable for CodeCR 567**
- 6.2.2.3.23. **Class Events – Not applicable for CodeCR 567**
- 6.2.2.3.24. **Class Methods – Not applicable for CodeCR 567**
- 6.2.2.3.25. **Class Properties – Not applicable for CodeCR 567**
- 6.2.2.3.26. **Uses Clause – Not applicable for CodeCR 567**
- 6.2.2.3.27. **Forms**

Table 46: Forms

Forms	Description
Form Name	Main.dfm
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Form Functionality	This is the main form for the CAPRI user interface. The Main Menu for CAPRI is on this form, and there is a new menu item.

Current Form Layout
See 6.2.2.3.17.

Modified Form Layout (Changes are in bold)
See 6.2.2.3.18. where “Get Docs From VLER DAS” is included and has a box around it.

6.2.2.3.28. **Functions**

Table 49: Functions

Function Name	actFileRetrieveDocsVLERDASUpdate
Short Description	New routine to disable Retrieve from VLER DAS menu item when no VLER DAS URL is available.
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
N/A		

Function Name	actFileRetrieveDocsVLERDASUpdate
---------------	----------------------------------

Function Name	actFileRetrieveDocsVLERDASUpdate
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition:
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic
N/A

Modified Logic (Changes are in bold)
<pre> {===== actFileRetrieveDocsVLERDASUpdate This action handler disables Retrieve from VLER DAS when no VLER DAS URL is available. CodeCR567 - Security Keys JRL 05/18/14 =====} procedure TfrmMain.actFileRetrieveDocsVLERDASUpdate(Sender: TObject); begin // Update Execute method for "Get Docs from VLER DAS" if (FVlerDasURL = '') then actFileRetrieveDocsVLERDAS.Enabled := False else actFileRetrieveDocsVLERDAS.Enabled := True; end;</pre>

Function Name	actFileConnectExecute
Short Description	Add a security key for returning docs from VLER DAS menu items.

Function Name	actFileConnectExecute
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
N/A		

Function Name	actFileConnectExecute
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition:
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic

Current Logic

```

If UserKeys.Count > 0 Then
Begin
    ReadOnlyMode : IsUserKeyInList('DVBA CAPRI READ ONLY,DVBA CAPRI READ O');
    actToolsExamListParameters.Visible : IsUserKeyInList('DVBA CAPRI EXAM LIST EDIT,DVBA CAPRI EXAM LIST E,DVBA C
SUPERVISOR');
    If IsUserKeyInList('DVBA CAPRI WORKSHEET TAB,DVBA CAPRI WORKSHEET ') Then
        Begin
            TabCPWorksheets.Visible : True;
            TabCPWorksheets.Enabled : True;
            TabCPWorksheets.TabVisible : True;
            actToolsUnsignedWorksheets.Visible : True;
        End;
        // Check Patch Number for defensive coding
        if IsPatchInstalled('DVBA*2.7*181') then                                // CodeCR347    JRL 6/12/12
        begin                                                                    // CodeCR347    JRL 6/12/12
            // Make the vocational rehab tab visible only if the user                // CodeCR347    JRL 6/12/12
            // has VocRehab userkeys                                                // CodeCR347    JRL 6/12/12
6/12/12    TabVocRehab.TabVisible : IsUserKeyInList('DVBA CAPRI VRE_COUNSELOR,DVBA CAPRI VRE_COUNSELOR'); // CodeCR347    JRL
            if TabVocRehab.TabVisible then                                        // CodeCR347    JRL 6/12/12
            begin                                                                // CodeCR347    JRL 6/12/12
                if not Assigned(VocRehab) then                                    // CodeCR347    JRL 6/12/12
                    VocRehab : TVocRehab.Create;                                // CodeCR347    JRL 6/12/12
                // assign access if user has VHA VocRehab privileges                // CodeCR347    JRL 6/12/12
                // (used for assigning consults)                                    // CodeCR347    JRL 6/12/12
6/12/12    VocRehab.VHAAccess : IsUserKeyInList('DVBA CAPRI VHA_COORDINATOR,DVBA CAPRI VHA_COORDINATOR'); // CodeCR347    JRL
            end;                                                                // CodeCR347    JRL 6/12/12
        end                                                                    // CodeCR347    JRL 6/12/12
        else                                                                    // CodeCR347    JRL 6/12/12
            TabVocRehab.TabVisible : FALSE;                                     // CodeCR347    JRL 6/12/12
            // DVBA CAPRI DENY GETVBADOCs was added in patch 186. VBA users could
            // be assigned this key. If this key was assigned, then the menu option
            // 'Get Docs from VVA' would not be visible. After implementation, this
            // was determined to be hard to administrate, so this key is being
            // deprecated and a new key will be added in patch 187 to all users
            // to allow them access to 'Get Docs from VVA'. VBA users not allowed
            // to see VVA docs here will not get this key. JRL 5/15/14
            if IsUserKeyInList('DVBA CAPRI DENY_GETVBADOCs') then                // CodeCR497    JRL 6/07/13
                actFileRetrieveVirtualVA.Visible : FALSE;                        // CodeCR497    JRL 6/07/13
            FoundCPWMKey : '';
            If IsUserKeyInList('DVBAB CPWM REQUIRE REVIEW,DVBAB CPWM REQUIRE REV') Then
                FoundCPWMKey : FoundCPWMKey + 'DVBAB CPWM REQUIRE REVIEW,';
            If IsUserKeyInList('DVBAB CPWM DISALLOW REVIEW,DVBAB CPWM DISALLOW REV') Then
                FoundCPWMKey : FoundCPWMKey + 'DVBAB CPWM DISALLOW REVIEW,';
            If IsUserKeyInList('DVBAB CPWM OPTIONAL REVIEW,DVBAB CPWM OPTIONAL REV') Then
                FoundCPWMKey : FoundCPWMKey + 'DVBAB CPWM OPTIONAL REVIEW,';
            boReviewerKey : IsUserKeyInList('DVBAB CPWM REVIEWER,DVBAB CPWM REVIE');

```

Modified Logic (Changes are in bold)

Modified Logic (Changes are in bold)

```

If UserKeys.Count > 0 Then
Begin
    ReadOnlyMode : IsUserKeyInList('DVBA CAPRI READ ONLY,DVBA CAPRI READ O');
    actToolsExamListParameters.Visible : IsUserKeyInList('DVBA CAPRI EXAM LIST EDIT,DVBA CAPRI EXAM LIST E,DVBA C
SUPERVISOR');
    If IsUserKeyInList('DVBA CAPRI WORKSHEET TAB,DVBA CAPRI WORKSHEET ') Then
        Begin
            TabCPWorksheets.Visible : True;
            TabCPWorksheets.Enabled : True;
            TabCPWorksheets.TabVisible : True;
            actToolsUnsignedWorksheets.Visible : True;
        End;
        // Check Patch Number for defensive coding
        if IsPatchInstalled('DVBA*2.7*181') then // CodeCR347 JRL 6/12/12
        begin // CodeCR347 JRL 6/12/12
            // Make the vocational rehab tab visible only if the user // CodeCR347 JRL 6/12/12
            // has VocRehab userkeys // CodeCR347 JRL 6/12/12
            TabVocRehab.TabVisible : IsUserKeyInList('DVBA CAPRI VRE_COUNSELOR,DVBA CAPRI VRE_COUNSELOR'); // CodeCR347 JRL
6/12/12
            if TabVocRehab.TabVisible then // CodeCR347 JRL 6/12/12
            begin // CodeCR347 JRL 6/12/12
                if not Assigned(VocRehab) then // CodeCR347 JRL 6/12/12
                    VocRehab : TVocRehab.Create; // CodeCR347 JRL 6/12/12
                // assign access if user has VHA VocRehab privileges // CodeCR347 JRL 6/12/12
                // (used for assigning consults) // CodeCR347 JRL 6/12/12
                VocRehab.VHAAccess : IsUserKeyInList('DVBA CAPRI VHA_COORDINATOR,DVBA CAPRI VHA_COORDINATOR'); // CodeCR347 JRL
6/12/12
            end; // CodeCR347 JRL 6/12/12
        end // CodeCR347 JRL 6/12/12
        else // CodeCR347 JRL 6/12/12
            TabVocRehab.TabVisible : FALSE; // CodeCR347 JRL 6/12/12
            // DVBA CAPRI DENY GETVBADOCs was added in patch 186. VBA users could
            // be assigned this key. If this key was assigned, then the menu option
            // 'Get Docs from VVA' would not be visible. After implementation, this
            // was determined to be hard to administrate, so this key is being
            // deprecated and a new key will be added in patch 187 to all users
            // to allow them access to 'Get Docs from VVA'. VBA users not allowed
            // to see VVA docs here will not get this key. JRL 5/15/14
            if IsUserKeyInList('DVBA CAPRI DENY_GETVBADOCs') then // CodeCR497 JRL 6/07/13
                actFileRetrieveVirtualVA.Visible : FALSE // CodeCR497 JRL 6/07/13
            // Add in security key for retrieving documents from VLER/DAS // CodeCR567 - JRL 5/22/14
            // You must have this key to see the menu option Get Docs from VLER // CodeCR567 - JRL 5/22/14
            if IsPatchInstalled('DVBA*2.7*187') then // CodeCR567 - JRL 5/22/14
            begin // CodeCR567 - JRL 5/22/14
                if IsUserKeyInList('DVBA CAPRI GETDOCSFROMVLER') then // CodeCR567 - JRL 5/18/14
                begin // CodeCR567 - JRL 5/22/14
                    actFileRetrieveDocsVLERDAS.Visible : TRUE; // CodeCR567 - JRL 5/18/14
                    actFileSeparatorBarVisible.Visible : TRUE; // CodeCR567 - JRL 5/22/14
                end // CodeCR567 - JRL 5/22/14
                else // CodeCR567 - JRL 5/18/14
                begin // CodeCR567 - JRL 5/22/14
                    actFileRetrieveDocsVLERDAS.Visible : FALSE; // CodeCR567 - JRL 5/18/14
                    actFileSeparatorBarVisible.Visible : TRUE; // CodeCR567 - JRL 5/22/14
                end // CodeCR567 - JRL 5/22/14
            end; // CodeCR567 - JRL 5/22/14
            FoundCPWMKey : '';
            If IsUserKeyInList('DVBAB CPWM REQUIRE REVIEW,DVBAB CPWM REQUIRE REV') Then
                FoundCPWMKey : FoundCPWMKey + 'DVBAB CPWM REQUIRE REVIEW,';
            If IsUserKeyInList('DVBAB CPWM DISALLOW REVIEW,DVBAB CPWM DISALLOW REV') Then
                FoundCPWMKey : FoundCPWMKey + 'DVBAB CPWM DISALLOW REVIEW,';
            If IsUserKeyInList('DVBAB CPWM OPTIONAL REVIEW,DVBAB CPWM OPTIONAL REV') Then
                FoundCPWMKey : FoundCPWMKey + 'DVBAB CPWM OPTIONAL REVIEW,';
            boReviewerKey : IsUserKeyInList('DVBAB CPWM REVIEWER,DVBAB CPWM REVIE');

```

- 6.2.2.3.29. Dialog – Not applicable for CodeCR 567
- 6.2.2.3.30. Help Frame – Not applicable for CodeCR 567
- 6.2.2.3.31. HL7 Application Parameter – Not applicable for CodeCR 567
- 6.2.2.3.32. HL7 Logical Link – Not applicable for CodeCR 567
- 6.2.2.3.33. COTS Interface – Not applicable for CodeCR 567
- 6.2.2.4. CODECR566 – Get Docs from VLER DAS
 - 6.2.2.4.1. Routines (Entry Points) - Not applicable for CodeCR 566
 - 6.2.2.4.2. Templates - Not applicable for CodeCR 566
 - 6.2.2.4.3. Bulletins - Not applicable for CodeCR 566
 - 6.2.2.4.4. Data Entries Affected by the Design - Not applicable for CodeCR 566
 - 6.2.2.4.5. Unique Record(s) - Not applicable for CodeCR 566
 - 6.2.2.4.6. File or Global Size Changes - Not applicable for CodeCR 566
 - 6.2.2.4.7. Mail Groups - Not applicable for CodeCR 566
 - 6.2.2.4.8. Security Keys

See section 6.2.2.3 VLER DAS Security Keys.
 - 6.2.2.4.9. Options - Not applicable for CodeCR 566
 - 6.2.2.4.10. Protocols - Not applicable for CodeCR 566
 - 6.2.2.4.11. Remote Procedure Call (RPC) - Not applicable for CodeCR 566
 - 6.2.2.4.12. Constants Defined in Interface - Not applicable for CodeCR 566
 - 6.2.2.4.13. Variables Defined in Interface - Not applicable for CodeCR 566
 - 6.2.2.4.14. Types Defined in Interface - Not applicable for CodeCR 566
 - 6.2.2.4.15. GUI

Table 36: GUI

Unit Name	Description
frmVlerGetExams.pas	Allows the user to input search criteria to retrieve Exams documents from VLER DAS.

6.2.2.4.16. GUI Classes - Not applicable for CodeCR566

6.2.2.4.17. Current Form

The screenshot shows a Windows-style application window titled "Retrieve Exam". Inside the window, there is a section titled "Exam Search Criteria" with a green background. This section contains several input fields: "Patient SSN" and "VA Facility ID" (both text boxes), "Physician First Name", "Middle Name", and "Last Name" (all text boxes), and "Exam Status" (a dropdown menu). Below these are "Creation Start Date:" and "Creation End Date:" labels, each followed by a date picker showing "6/ 9/2014". At the bottom of the criteria section are three buttons: "Search", "Clear Fields", and "Close". An XML icon is visible in the top right corner of the criteria section.

6.2.2.4.18. Modified Form- Not applicable for CodeCR566

6.2.2.4.19. Components on Form

Table 39: Components on Form

Name	Type	Description
edtTPatientSSN	TEdit	Enter SSN number
cmbxFcltyLoc	TCombox	Enter a VA Facility Number
cmbxPhysicianFnm	TCombox	Enter Physician First Name
cmbxPhysicianMnm	TCombox	Enter Physician Middle Name
cmbxPhysicianLnm	TCombox	Enter Physician Last Name
cmbxExmStatus	TCombox	Choose a Exam Status
dtmepckrCreateStrtDT	TDateTimePicker	Select/Type Start Date Range
dtmepckrCreateEndDT	TDateTimePicker	Select/Type End Date Range
btnMultiSrch	TButton	Starts Search for Exams
btnClearFields	TButton	Clears all search fileds
btnClose	TButton	Close from frmVlerGetExam

6.2.2.4.20. Events

Table 40: Events

Name	Type	Description
edtTPatientSSNExit	onExit	Validates the value entered for interger, length of 9 or 10, if length of 10 and 10 position is 'P' the allow .
dtmepckrCreateStrtDTExit	onExit	Validates Start Date is not greater than Creation End Date
dtmepckrCreateEndDTExit	onExit	Validates Creation End Date is not less than Creation Start Date.
btnMultiSrchClick	onClick	Initiates start search.
btnClearFieldsClick	onClick	Clear all search fields.

6.2.2.4.21. Methods

Table 41: Methods

Method Name	Procedure/Function	Description
FormDestroy	Procedure	Frees objects strlstFacilityandDateRange, strlstPhysicianDateRange, strlstExamStatusDateRange, and strlstMultiSelect.
setSearchExamparms	Procedure	Set field values for Type record TVLERGetDocSearch
convertDateToGMTStr	Function	Convert dtmepckrCreateStrtDT to GMT format yyyy-mm-dd"T00:00:00-00:00. Convert dtmepckrCreateEndDT to yyyy-mm-dd"T23:59:59-00:00
setExamStatus	Procedure	Add default value to cmbxExmStatus
FreeTList	Function	Used to clear all assigned values to a TList.

6.2.2.4.22. Special References

Special Reference Name	Type	Description
TVLERGetDocSearch	Record	Holds Exam search results values

6.2.2.4.23. Class Events - Not applicable for CodeCR566

6.2.2.4.24. Class Methods - Not applicable for CodeCR566

6.2.2.4.25. Class Properties - Not applicable for CodeCR566

6.2.2.4.26. Uses Clause

interface

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, ComCtrls, StdCtrls, Grids, Mask, xmldom, XMLIntf, msxmldom,
XMLDoc, IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient, IdHTTP,
Diaccess, Fmcmpnts, Fmctrls, TRPCB, DialogsCAPRI;

implementation

Main,
CCOWRPCBrokerCAPRI,
eCrud,
frmVlerGetExamDocs;

6.2.2.4.27. Forms

Table 46: Forms

Forms	Description
Form Name	frmVlerGetExam
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Form Functionality	This form allows users to search and retrieve Exams from VLER DAS

Current Form Layout
N/A

Modified Form Layout (Changes are in bold)

Modified Form Layout (Changes are in bold)

Retrieve Exam

Exam Search Criteria

Patient SSN:

VA Facility ID:

Physician First Name:

Middle Name:

Last Name:

Exam Status:

Creation Start Date:

Creation End Date:

XML

6.2.2.4.28. Functions

Table 48: Functions

Function Name - setSearchExamparms	Activities
Short Description	Set field values for Type record TVLERGetDocSearch
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	

Related Routines	Routines "Called By"	Routines "Called"
	procedure TfrmVlerGetExams.setSearchExamparms;	

Function Name	Activities
Data Dictionary (DD) References	N/A
Related Protocols	N/A

Function Name	Activities
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input checked="" type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: InDate: TDateTime; Definition: Date to convert to GMT
Output Attribute Name and Definition	Name: StartDt : Boolean Definition: Used to determine if start or end date is the first parameter.

Current Logic
<pre>function TfrmVlerGetExams.convertDateToGMTStr(InDate: TDateTime; StartDt : Boolean): String; var GMT : String; begin // If you put in the same day for date range, the start time should be at 00:00:00 // and the end time should be 23:59:59 in order to return any results if StartDt then GMT: FormatDateTime('yyyy-mm-dd"T00:00:00-00:00"', InDate) else GMT: FormatDateTime('yyyy-mm-dd"T23:59:59-00:00"', InDate); result: GMT; end;</pre>

Modified Logic (Changes are in bold)

6.2.2.4.29.

Table 48: Forms

Function Name – Procedure setSearchExamparms	Activities
Short Description	Set field values for Type record TVLERGetDocSearch
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	

Related Routines	Routines “Called By”	Routines “Called”
	procedure TfrmVlerGetExams.btnMultiSrchClick(Sender: TObject);	

Function Name	Activities
Data Dictionary (DD) References	N/A
Related Protocols	N/a
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition:
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic

Current Logic

```
procedure TfrmVlerGetExams.setSearchExamparams;
begin

    if Trim(edtTPatientSSN.Text) <> '' then
        GetExamParams.strPatientId :  edtTPatientSSN.Text
    else
        GetExamParams.strPatientId :  IGNORE;

    if Trim(cmbxFcltyLoc.Text) <> '' then
        GetExamParams.strfacilityID :  cmbxFcltyLoc.Text
    else
        GetExamParams.strfacilityID :  IGNORE;

    if Trim(cmbxPhysicianFnm.Text) <> '' then
        GetExamParams.strPhysicianFnm :  cmbxPhysicianFnm.Text
    else
        GetExamParams.strPhysicianFnm :  IGNORE;

    if Trim(cmbxPhysicianMnm.Text) <> '' then
        GetExamParams.strPhysicianMnm :  cmbxPhysicianMnm.Text
    else
        GetExamParams.strPhysicianMnm :  IGNORE;

    if Trim(cmbxPhysicianLnm.Text) <> '' then
        GetExamParams.strPhysicianLnm :  cmbxPhysicianLnm.Text
    else
        GetExamParams.strPhysicianLnm :  IGNORE;

    if Trim(cmbxExmStatus.Text) <> '' then
        GetExamParams.strExamstatus :  cmbxExmStatus.Text
    else
        GetExamParams.strExamstatus :  IGNORE;

    if Trim(DateToStr(dttmepckrCreateStrtDT.DateTime)) <> '' then
        GetExamParams.strCreationStartDT :  convertDateToGMTStr(dttmepckrCreateStrtDT.DateTime,TRUE)
    else
        GetExamParams.strCreationStartDT :  IGNORE;

    if Trim(DateToStr(dttmepckrCreateEndDT.DateTime)) <> '' then
        GetExamParams.strCreationEndDt :  convertDateToGMTStr(dttmepckrCreateEndDT.DateTime,FALSE)
    else
        GetExamParams.strCreationEndDt :  IGNORE;

    (* showmessage('GetExamParams: ' +  GetExamParams.strPatientId + ' : ' +
        GetExamParams.strPhysicianFnm + ' : ' +
        GetExamParams.strPhysicianMnm + ' : ' +
        GetExamParams.strPhysicianLnm + ' : '+GetExamParams.strfacilityID + ' : ' +
        GetExamParams.strExamstatus + ' : ' + GetExamParams.strCreationStartDT +
        ' : ' + GetExamParams.strCreationEndDt); *)

end;
```

Table 36: GUI

Unit Name	Description
frmVlerGetExamDocs	Displays results from search criteria from form frmVlerGetExams. Then the use can very the binary attachment(the actual exam reports/images)

6.2.2.4.30. GUI Classes - Not applicable for CodeCR566

6.2.2.4.31. Current Form

6.2.2.4.32. Modified Form- Not applicable for CodeCR566

6.2.2.4.33. Components on Form

Table 39: Components on Form

Name	Type	Description
edbPatientName	TEdit	Display Patient Full Name
edbClaimNum	TEdit	Display Patient SSN
edbDOB	TEdit	Display Patient DOB
cbSortColumn	TCombox	Allows user to select column to sort on.
cbSortDirection	TCombox	Allows user to select sort order

Name	Type	Description
btnViewSelectedDocument	TButton	Get associate Binary Attachment
strgrdCollectResults	TStringGrid	Contain search result from frmVlerGetExam
btnClose	TButton	Close from frmVlerGetExamDocs

6.2.2.4.34. Events

Table 40: Events

Name	Type	Description
cbSortColumnCloseUp	onCloseup	Executes sort of grid
cbSortDirectionCloseUp	onCloseup	Set sort oder
btnViewSelectedDocumentClick	onClick	Retrieves binary attachment from VLER and opens file based on filename suffix.
strgrdCollectResultsClick	onClick	Refreshes Patient information group box
strgrdCollectResultsDbClick	onDoubleClick	<ul style="list-style-type: none"> - Retrieves binary attachment from VLER and opens file based on filename suffix. - Refreshes Patient information group box

6.2.2.4.35. Methods

Table 41: Methods

Method Name	Procedure/Function	Description
PatientHeader	Procedure	Refreshes Patient information group box
AssignSortColumns	Procedure	Populates cbSortColumn
SortStringgridAndOrder	Procedure	Implements the sorting of strgridCollectionResultss
ExchangeGridRows	Procedure	Helper procedure called by procedure SortStringgridAndOrder
QuickSort	Procedure	Helper procedure called by procedure SortStringgridAndOrder

Method Name	Procedure/Function	Description
InvertGrid	Procedure	Helper procedure called by procedure SortStringgridAndOrder

6.2.2.4.36. Special References - Not applicable for CodeCR566

6.2.2.4.37. Class Events - Not applicable for CodeCR566

6.2.2.4.38. Class Methods - Not applicable for CodeCR566

6.2.2.4.39. Class Properties - Not applicable for CodeCR566

6.2.2.4.40. Uses Clause

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, VA508AccessibilityManager, Grids, ShellAPI, eCrud, XMLDoc, IdHTTP, xmldom, XMLIntf, IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient, msxmldom;

implementation

Main, CCOWRPCBrokerCAPRI, eCrud, frmVlerGetExamDocs;

6.2.2.4.41. Forms

Table 46: Forms

Forms	Description
Form Name	frmVlerGetExamDocs
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Form Functionality	This form allows users to view retrieved Exams from VLER DAS and open binary attachments

Current Form Layout
N/A

Modified Form Layout (Changes are in bold)
--

Modified Form Layout (Changes are in bold)

Retrieve Exam Documents from VLER

Patient

Name: SSN: DOB:

Sort By

Sort Column Sort Direction

[View Selected Document](#)

Document List:

Close

6.2.2.4.42. Functions

Table 48: Functions

Function Name -	Activities
Short Description	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	N/A	N/A

Function Name	Activities
Data Dictionary (DD) References	N/A

Function Name	Activities
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
	Name: N/A Definition:
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic

Modified Logic (Changes are in bold)

6.2.2.4.43. Class Events

Table 42: Class

Name	Type	Description
eCrud.pas	Library	Used in frmVlerGetExams and frmVlerGetExamDocs

6.2.2.4.44. Class Methods

Table 43: Class Methods

Name	Procedure/Function	Description
getExamReviewDocument	Function	Executes query to VLER Das and extracts information from XML result to populate frmVlerGetExamDocs.
SelectNodes	Function	
NodeFound	Function	Validates a XML node exists
getExamBy	Function	Creates URL queries based on users search selection values provided on form frmVlerGetExams. Physician search matches name of person id:two only. Searches are filtered to pass only Status Completed or Rejected. Then the URL string is passed to function getExamReviewDocuments().
getExamAttachments	Procedure	Is executed from frmVlerGetExamDocs form when the View button is clicked. Create a URI query.
getExamAttachmentList	Procedure	Assign pointers
OpenDocumentFromXMLNode	Procedure	Execute a URL query using the URL string created in procedure getExamAttachments, save the binary result to a file on user local temp directory. Then the file is open based on the filename suffix using procedure OpenDocument defined in unitDocumentMethods.pas.

Name	Procedure/Function	Description
CloseOpenAttachments	Procedure	Is call to close all open Binary documents opened by procedure OpenDocumentFromXMLNode(). This procedure is found in UnitdocumentMethods.
GetTempDirectory	Function	Gets the users local temporary directory path.

6.2.2.4.45. Class Properties

Table 43: Class Properties

Name	Type	Description
XMLDocument	Property	Allows the assignment of and access to private FXMLDocument.
eCRUDVersion	Property	Allows the assignment of and access to private FeCRUDVersion. Interface version number.
dbName	Property	Allows the assignment of and access to private FdbName. The VLER Data Store (VDS) MongoDB database name where the data for this operation is to be read
collectionName	Property	Allows the assignment of and access to private FcollectionName. for MongoDB read requests: this is the VDS, MongoDB collection name, for GridFS partition, this will always have the value of "fs".

Name	Type	Description
maxRecords	Property	<p>Allows the assignment of and access to private FmaxRecords.</p> <p>the optional "limit=" request integer parameter, accepts a {maxRecords} value which sets the maximum number of results that can be sent back in the response.</p> <p>When not specified: will default to 100 max records, when zero: implies no limits and all records will be returned and when negative: the absolute value will be considered.</p>
SpecifyResponseFields	Property	<p>Allows the assignment of and access to private FSpecifyResponseFields.</p> <p>The optional "fields=" request parameter, accepts a {columnProjections} value which sets the fields that can be included/not included in the response. When not specified will return all fields in the document</p>
Sort	Property	<p>Allows the assignment of and access to private FSort.</p>

- 6.2.2.4.46. Dialog - Not applicable for CodeCR 566
- 6.2.2.4.47. Help Frame - Not applicable for CodeCR 566
- 6.2.2.4.48. HL7 Application Parameter - Not applicable for CodeCR 566
- 6.2.2.4.49. HL7 Logical Link - Not applicable for CodeCR 566
- 6.2.2.4.50. COTS Interface - Not applicable for CodeCR 566
- 6.2.2.5. CODECR565 – Separate Current Merged PDFs
 - 6.2.2.5.1. Routines (Entry Points) - Not applicable for CodeCR565
 - 6.2.2.5.2. Templates - Not applicable for CodeCR565
 - 6.2.2.5.3. Bulletins - Not applicable for CodeCR565
 - 6.2.2.5.4. Data Entries Affected by the Design - Not applicable for CodeCR565
 - 6.2.2.5.5. Unique Record(s) - Not applicable for CodeCR565
 - 6.2.2.5.6. File or Global Size Changes - Not applicable for CodeCR565
 - 6.2.2.5.7. Mail Groups - Not applicable for CodeCR565
 - 6.2.2.5.8. Security Keys - Not applicable for CodeCR565
 - 6.2.2.5.9. Options - Not applicable for CodeCR565
 - 6.2.2.5.10. Protocols - Not applicable for CodeCR565
 - 6.2.2.5.11. Remote Procedure Call (RPC) - Not applicable for CodeCR565
 - 6.2.2.5.12. Constants Defined in Interface - Not applicable for CodeCR565
 - 6.2.2.5.13. Variables Defined in Interface - Not applicable for CodeCR565
 - 6.2.2.5.14. Types Defined in Interface - Not applicable for CodeCR565
 - 6.2.2.5.15. GUI

Table 36: GUI

Unit Name	Description
PNCMain.pas, TIUSign.pas, SplitExamInfo.pas	Alter the units mentioned to take the currently “merged” PDF sent to both Virtual VA and VLER DAS and split it out into a single PDF per exam.

- 6.2.2.5.16. GUI Classes - Not applicable for CodeCR565
- 6.2.2.5.17. GUI Classes

Table 38: GUI Classes

GUI Classes	Instructions
Class Name	TSplitExam
Derived From Class	N/A

GUI Classes	Instructions
Purpose	This class contains routines specifically to support unmerging the PDF file that is sent to Virtual VA. Virtual VA will still receive a merged PDF, but the DBQs sent to VLER/DAS will now have XML and the PDF associated with that XML only.

6.2.2.5.18. Current Form - Not applicable for CodeCR565

6.2.2.5.19. Modified Form - Not applicable for CodeCR565

6.2.2.5.20. Components on Form - Not applicable for CodeCR565

6.2.2.5.21. Events - Not applicable for CodeCR565

6.2.2.5.22. Methods - Not applicable for CodeCR565

6.2.2.5.23. Special References - Not applicable for CodeCR565

6.2.2.5.24. Class Events - Not applicable for CodeCR565

6.2.2.5.25. Class Methods

Table 43: Class Methods

Name	Procedure/Function	Description
RavePrint	Procedure	Report Print routine
RavePrintFooter	Procedure	Report Print footer routine
RavePrintHeader	Procedure	Report Print header routine
RaveBeforePrint	Procedure	Report Print routine
PrintLine	Procedure	Report Print routine
GetFileSize	Function	Get File Size routine
ReplaceTabs	Function	Report Print routine
CreateXmitFile	Function	Cereate file for transmission to VLER
GetServerDateTime	Function	Get VistA server date and time
CompressPDF	Procedure	Compress and linearize PDF for smaller transmissions
PadString	Function	Report Print routine
ProcessTextToPDF	Function	Create PDF files from text stored in memory

6.2.2.5.26. Class Properties - Not applicable for CodeCR565

6.2.2.5.27. Uses Clause

interface uses:

Windows, SysUtils, Classes, Dialogs, DateUtils, CommCtrl, Types, Graphics, Controls, StrUtils, RpRender, RpRenderPDF, RpDefine, RpBase, RpSystem, ShellAPI, ComCtrls, Forms, DialogsCAPRI, CAPRISupport;

Implementation uses:

Main, trpcb, untMiscMthds;

6.2.2.5.28. Forms - Not applicable for CodeCR565

6.2.2.5.29. Functions

Table 48: Functions

Function Name	RavePrint
Short Description	Report Print routine
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	CreateXmitFile	PrintLine

Function Name	RavePrint
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name:N/A Definition: N/A
Output Attribute Name and Definition	Name:N/A Definition: N/A

Current Logic

N/A

Modified Logic (Changes are in bold)

```

=====
RavePrint
This procedure is the Rave Reports OnPrint handler that breaks the
report text into pages.
=====
procedure TSplitExam.RavePrint(Sender: TObject);
var
  I: Integer;
  currentLineHeight: Double;
  footerLineHeight: Double;
begin
  with Sender as TBaseReport do begin
    footerLineHeight := 0.2;
    SetFont(vlerReport.FontName, vlerReport.FontSize);
    for I := 0 to vlerReport.ReportText.Count - 1 do
      begin
        PrintLine(TBaseReport(Sender), vlerReport.ReportText[I]);
        currentLineHeight := LineHeight;
        LineHeight := footerLineHeight;
        if (LinesLeft < 5) then //need 4 lines for footer
          NewPage;
        LineHeight := currentLineHeight;
      end;
    end;
  end;
end;
=====

```

Table 48: Functions

Function Name	RavePrintHeader
Short Description	Report Print header routine
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	CreateXmitFile	N/A

Function Name	RavePrintHeader
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference

Function Name	RavePrintHeader
Input Attribute Name and Definition	Name: N/A Definition: N/A
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic

N/A

Modified Logic (Changes are in bold)

```

=====
RavePrintHeader
This procedure is the placeholder for the Rave Reports OnPrintHeader
handler. Not used at this time.
=====
}
procedure TSplitExam.RavePrintHeader(Sender: TObject);
begin
    //Not currently used    placeholder only.
end;

```

Table 48: Functions

Function Name	RavePrintFooter
Short Description	Report Print footer routine
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	CreateXmitFile	N/A

Function Name	RavePrintFooter
Data Dictionary (DD) References	N/A
Related Protocols	N/A

Function Name	RavePrintFooter
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition: N/A
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic

N/A

Modified Logic (Changes are in bold)

```

=====
RavePrintFooter
This procedure is the Rave Reports OnPrintFooter handler that
formats the report footer and occurs after each NewPage call.
=====
procedure TSplitExam.RavePrintFooter(Sender: TObject);
var
  X, Y: double;
  SavedPen, LinePen: TPen;
begin
  with Sender as TBaseReport do begin
    SaveFont(1);
    SetFont('Arial', 10);
    SectionBottom := SectionBottom    (3 * LineHeight);
    GoToFooter;

    //Draw separator line
    SavedPen := Canvas.Pen;
    LinePen := CreatePen(clBlack, psSolid, 10, pmBlack);
    try
      Canvas.Pen := LinePen;
      X := XD2I(CursorXPos);
      Y := YD2I(CursorYPos);
      MoveTo(X, Y);
      LineTo(SectionRight, Y);
    finally
      Canvas.Pen := SavedPen;
      FreeAndNil(LinePen);
    end;

    //footer line 1
    SectionBottom := SectionBottom + LineHeight;
    PrintFooter(vlerReport.FooterDesc, pjLeft);
    PrintFooter('Page: ' + IntToStr(CurrentPage), pjRight);

    //footer line 2
    SectionBottom := SectionBottom + LineHeight;
    PrintFooter(vlerReport.Patient.FullName + '    SSN#' + vlerReport.Patient.SSN, pjLeft);
    PrintFooter('Printed on: ' + vlerReport.PrintDate, pjRight);

    //footer line 3
    SectionBottom := SectionBottom + LineHeight;

```

```

PrintFooter('System: ' + vlerReport.SystemName, pjLeft);
PrintFooter('Division: ' + vlerReport.Division, pjRight);

RestoreFont(1);
end;
end;

```

Table 48: Functions

Function Name	RaveBeforePrint
Short Description	Report Print routine
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	N/A

Function Name	RaveBeforePrint
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition: N/A
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic

N/A

Modified Logic (Changes are in bold)

```

=====
RaveBeforePrint
This procedure is the Rave Reports OnBeforePrint handler that provides
a setup location for properties prior to the OnPrint event.
=====
}
procedure TSplitExam.RaveBeforePrint(Sender: TObject);
begin
  with Sender as TBaseReport do
  begin
    LineHeight := 0.147; //just to match existing QuickReports spacing
  end;
end;

```

Table 48: Functions

Function Name	PrintLine
Short Description	Report Print routine
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	RavePrint	

Function Name	PrintLine
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: ReportComponent, ReportLine Definition: TBaseReport, String
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic

N/A

Modified Logic (Changes are in bold)

```
{=====
PrintLine
This procedure provides a wrapper around the TBaseReport PrintLn procedure
to prevent Tab (#9) characters from crashing the report writer.
=====}
procedure TSplitExam.PrintLine(ReportComponent: TBaseReport;
  ReportLine: String);
var
  cleanLine, trimmedLine: String;
  printWidth : Double;
  maxChars: Integer;
begin
  if Pos(#9, ReportLine) > 0 then
    cleanLine := ReplaceTabs(ReportLine, 6)
  else
    cleanLine := ReportLine;

  {Some reports erroneously return long strings of periods '.' that
  are intended as a keepalive for interactive terminal sessions.
  These strings cause errors when they are longer than 255 characters.
  The following code attempts to prevent lines of periods from printing and
  also wraps the remaining valid text that is longer than the available
  page print width.}
  printWidth := ReportComponent.PageWidth - ReportComponent.MarginLeft - ReportComponent.MarginRight;
  if ReportComponent.TextWidth(cleanLine) > printWidth then
    begin
      maxChars := Round(Length(cleanLine)/ReportComponent.TextWidth(cleanLine) * printWidth);
      while (Length(cleanLine) > maxChars) do
        begin
          trimmedLine := Copy(cleanLine, 0, maxChars);
          if trimmedLine <> StringOfChar('.', Length(trimmedLine)) then //only print if not periods
            ReportComponent.Println(trimmedLine);

          cleanLine := Copy(cleanLine, maxChars + 1, Length(cleanLine));
          if cleanLine = StringOfChar('.', Length(cleanLine)) then //clear "periods only" line
            cleanLine := '';
        end;
      end;

      ReportComponent.Println(cleanLine);
    end;
end;
```

Table 48: Functions

Function Name	GetFileSize
Short Description	Get File Size routine
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	CreateXmitFile	N/A

Function Name	GetFileSize
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input checked="" type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: Filename Definition: String
Output Attribute Name and Definition	Name: FileSize Definition: LongInt

Current Logic

N/A

Modified Logic (Changes are in bold)

```

=====
GetFileSize
This function determines the size of a given file and returns the size
in bytes.
=====
function TSplitExam.GetFileSize(const FileName: String): LongInt;
var
  aFile: File of Byte;
begin
  AssignFile(aFile, FileName);
  Reset(aFile);
  try
    Result := FileSize(aFile);
  finally
    Close(aFile);
  end;
end;

```

Table 48: Functions

Function Name	ReplaceTabs
Short Description	Report Print routine
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change

Function Name	ReplaceTabs
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	PrintLine	N/A

Function Name	ReplaceTabs
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input checked="" type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: LineWithTabs, TabSize Definition: String, Integer
Output Attribute Name and Definition	Name: LineWithSpaces Definition: String

Current Logic

N/A

Modified Logic (Changes are in bold)

```

=====
ReplaceTabs
This utility function supports the PrintLine procedure and replaces all
tab characters with spaces in a given string based on the given tab size.
=====
function TSplitExam.ReplaceTabs(LineWithTabs: String;
    tabSize: Integer): String;
var
    I, padLength : Integer;
    LineWithSpaces : String;
begin
    LineWithSpaces := '';
    for I := 1 to Length(LineWithTabs) do
    begin
        if LineWithTabs[i] = #9 then
        begin
            padLength := tabSize - (Length(LineWithSpaces) MOD tabSize);
            LineWithSpaces := LineWithSpaces + StringOfChar(' ', padLength);
        end
    end
end

```



```

else
  LineWithSpaces := LineWithSpaces + LineWithTabs[i];
end;
Result := LineWithSpaces;
end;

```

Table 48: Functions

Function Name	CreateXmitFile
Short Description	Format and create the PDF output file that will be transmitted to DAS.
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	ProcessTextToPDF	CompressPDF GetFileSize

Function Name	CreateXmitFiles
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: Definition:
Output Attribute Name and Definition	Name: Definition:

Current Logic -

N/A

Modified Logic (Changes are in bold)

```

=====
CreateXmitFile
This function uses Rave Reports to format and create the PDF output file that
will be transmitted to VLER DAS. The existing PDF is then compressed and
linearized prior to sending to VLER DAS.

Input: vlerReport: TvlerReport    report object
Output: function result    returns True on success;
                           otherwise returns False.
=====
function TSplitExam.CreateXmitFile(var vlerReport: TvlerReport): LongInt;
const
    FILE_EXTENT = '.PDF';
begin
    // Make sure that PDF is compressed before it is updated into memory!!! CodeCR565 JRL 7/13/14
    // Do we need a flag to delete or not delete individual PDFs?

    RaveSystem.OnPrint := RavePrint;
    RaveSystem.OnPrintHeader := RavePrintHeader;
    RaveSystem.OnPrintFooter := RavePrintFooter;
    RaveSystem.DefaultDest := rdFile;
    RaveSystem.DoNativeOutput := false;
    RaveSystem.RenderObject := RaveRenderPDF;
    RaveSystem.OutputFileName := vlerReport.vlerDoc.OutputFilePath + vlerReport.vlerDoc.OutputFileName;
    RaveSystem.SystemSetups := RaveSystem.SystemSetups [ssAllowSetup];
    RaveSystem.Execute;
    if FileExists(RaveSystem.OutputFileName) then
    begin
        // make a compressed version of the PDF here to send to VLER/DAS filename has a 'c' added at the start
        try
            CompressPDF(RaveSystem.OutputFileName, vlerReport.vlerDoc.OutputFilePath + 'c' +
vlerReport.vlerDoc.OutputFileName);
        except
            on E:Exception do
                begin // Copy uncompressed file to compressed filename so it can be sent if there was an error
compressing
                    if FileExists(vlerReport.vlerDoc.OutputFilePath + 'c' + vlerReport.vlerDoc.OutputFileName) then
                        DeleteFile(pchar(vlerReport.vlerDoc.OutputFilePath + 'c' + vlerReport.vlerDoc.OutputFileName));
                    if FileExists(RaveSystem.OutputFileName) then
                        MoveFile(pchar(RaveSystem.OutputFileName), pchar(vlerReport.vlerDoc.OutputFilePath + 'c' +
vlerReport.vlerDoc.OutputFileName));
                    end; // compress except
                end; // compress try
                Result := GetFileSize(vlerReport.vlerDoc.OutputFilePath + vlerReport.vlerDoc.OutputFileName); // Return
file size
            end
        else
        begin
            Result := 0;
        end;
    end;
end;

```

Table 48: Functions

Function Name	GetServerDateTime
Short Description	Get Vista server date and time routine.
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	ProcessTextToPDF	N/A

Function Name	GetServerDateTime
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input checked="" type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name:N/A Definition:N/A
Output Attribute Name and Definition	Name:Server Date and Time String Definition: String

Current Logic

N/A

Modified Logic (Changes are in bold>

```

=====
GetServerDateTime
This function retrieves the current date/time from the connected
remote system in "HH:MM:SS am/pm" format. Refactored from
actPrintFileExecute for reuse.

Input: none
Output:  returns formatted date/time on success;
         otherwise, returns empty string
=====}
function TSplitExam.GetServerDateTime(): String;
begin
  frmMain.RPCBroker1.RemoteProcedure := 'DVBAB DATETIME';
  try
    frmMain.RPCBroker1.Call;
  except
    on EBrokerError do
      begin
        FormatDateTime('mmm dd, yyyy hh:mm:ss', Date);
      end;
    end;
  if (frmMain.RPCBroker1.Results.Count > 0) then
    Result := frmMain.RPCBroker1.Results[0]
  else
    Result := '';
  end;
end;

```

Table 48: Functions

Function Name	CompressPDF
----------------------	--------------------

Function Name	CompressPDF
Short Description	Compress and linearize the PDF file to shorten transmission times.
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	CreateXmitFile	N/A

Function Name	CompressPDF
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input checked="" type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: Uncompressed Filename, Compressed Filename Definition: String, String
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic

N/A

Modified Logic (Changes are in bold)

```

=====
CompressPDF
This function uses an external tool, QPDF, to compress the Rave Report
output PDF file to a compressed and linearized format. This file will be
transmitted to the Virtual VA. QPDF must finish compressing the file before
exiting from this routine to avoid file errors in subsequent code.

Parameters:
  Input:  Uncompressed PDF filename
  Output: Compressed PDF filename

```

Required files for running QPDF:

```
qpdf.exe
qpdf13.dll
libgcc_s_dw2-1.dll
libstdc++6.dll
```

Created routine: CodeCR559 3/13/14 JRL

```
=====}
procedure TSplitExam.CompressPDF(UncompressedFilename, CompressedFilename: String);
var
  SEInfo: TShellExecuteInfo;
  ExitCode: DWORD;
  ExecuteFile, ParamString, StartInString : string;
begin
  ParamString := ' linearize "' + UncompressedFilename + '" "' + CompressedFilename + '"';
  ExecuteFile:='qpdf.exe';
  FillChar(SEInfo, SizeOf(SEInfo), 0) ;
  SEInfo.cbSize := SizeOf(TShellExecuteInfo) ;
  with SEInfo do
  begin
    fMask := SEE_MASK_NOCLOSEPROCESS;
    Wnd := Application.Handle;
    lpFile := PChar(ExecuteFile) ;
    lpParameters := PChar(ParamString) ;
    StartInString := ExtractFilePath(Application.ExeName);
    lpDirectory := PChar(StartInString);
    nShow := SW_HIDE;
  end;

  if ShellExecuteEx(@SEInfo) then
  begin
    repeat
      Application.ProcessMessages;
      GetExitCodeProcess(SEInfo.hProcess, ExitCode) ;
    until (ExitCode <> STILL_ACTIVE) or
      Application.Terminated;
  end;
end;
```

Table 48: Functions

Function Name	PadString
Short Description	Report Print routine
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	ProcessTextToPDF	N/A

Function Name	PadString
Data Dictionary (DD) References	N/A
Related Protocols	N/A

Function Name	PadString
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input checked="" type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: InputString, Pad character, Length of padding Definition: String, Char, Integer
Output Attribute Name and Definition	Name: Padded String Definition: String;

Current Logic

N/A

Modified Logic (Changes are in bold)

```

=====
RightPad
This procedure right pads header strings with spaces.
=====}
function TSplitExam.PadString(S: string; Ch: Char; Len: Integer): string;
var
  RestLen: Integer;
begin
  Result := S;
  RestLen := Len - Length(s);
  if RestLen < 1 then Exit;
  Result := S + StringOfChar(Ch, RestLen);
  // Result := StringOfChar(Ch, RestLen) + S;
end;

```

Table 48: Functions

Function Name	ProcessTextToPDF
Short Description	Take exam results stored in memory and save them to PDFs.
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	TTIUSignForm.SendDbqsToVlerAndVista	PadString

Function Name	ProcessTextToPDF
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition: N/A
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic – N/A

Modified Logic (Changes are in bold)

```
//
// ProcessTextToPDF
//
function TSplitExam.ProcessTextToPDF (eSignature, eTitle, eSignatureDateTime, examAuthor, examCosigner, title,
DateOfNote : String) : Boolean;
var
  x : Integer;
  len : Integer;

begin
  //GetTempPath(MAX_PATH, @tempFolder);
  vlerReport.FontName := 'Courier New';
  vlerReport.FontSize := 10;
  vlerReport.Patient.FullName := patientname;
  vlerReport.Patient.SSN := patientssn;
  vlerReport.PrintDate := GetServerDateTime();
  VlerReport.SystemName := Uppercase(frmMain.RPCBroker1.ANUStrServer);
  if CallRPC(frmMain.RPCBroker1, 'XWB GET VARIABLE VALUE', ['DUZ("2")'], nil, True) then
    vlerReport.Division := frmMain.RPCBroker1.Results[0];
  vlerReport.FooterDesc := 'CLIN DOC: Progress Note';
  vlerReport.ReportText := TStringList.Create;
  vlerReport.ReportText.Clear;
  vlerReport.hdrTitle := title;
  vlerReport.hdrDate := eSignatureDateTime;
  vlerReport.hdrAuthor := examAuthor;
  vlerReport.hdrCosigner := examCosigner;
  vlerReport.hdrUrgency := ' '; // where do we get this?
  vlerReport.DateOfNote := DateOfNote;

  for x := 0 to frmMain.NumExamTextForPDFs - 1 do
  begin
    //vlerReport.vlerDoc.OutputFilePath := StrPas(tempFolder); // ??? should this have a CAPRI subfolder?
    vlerReport.vlerDoc.OutputFilePath := GetTempDir; // DONE: // CodeCR??? LMS 2014 07 14
    consolidate/refactor all GetTempDir into one location, so all can benefit from Win 7 specific handling.

    vlerReport.vlerDoc.OutputFileName := 'Exam' + IntToStr(x) + PDF_EXTENT;
```


6.2.2.5.30. Dialog - Not applicable for CodeCR565

6.2.2.5.31. GUI

Table 36: GUI

Unit Name	Description
PNCMain.pas	Alter the units mentioned to take the currently “merged” PDF sent to both Virtual VA and VLER DAS and split it out into a single PDF per exam.

6.2.2.5.32. GUI Classes - Not applicable for CodeCR565

6.2.2.5.33. GUI Classes - - Not applicable for CodeCR565

6.2.2.5.34. Current Form - Not applicable for CodeCR565

6.2.2.5.35. Modified Form - Not applicable for CodeCR565

6.2.2.5.36. Components on Form - Not applicable for CodeCR565

6.2.2.5.37. Events

Table 40: Events

Name	Type	Description
xFormOutputOKClick	Procedure	Modify routine to save the split apart exam text into memory. This will be used to create PDFs later.

6.2.2.5.38. Methods

Table 41: Methods

Method Name	Procedure/Function	Description
SaveExamsToMemory	Procedure	Save the text on each tab (one exam result per tab) into a separate memory location to save in a PDF later.

- 6.2.2.5.39. **Special References - Not applicable for CodeCR565**
- 6.2.2.5.40. **Class Events - Not applicable for CodeCR565**
- 6.2.2.5.41. **Class Methods - Not applicable for CodeCR565**
- 6.2.2.5.42. **Class Properties - Not applicable for CodeCR565**
- 6.2.2.5.43. **Uses Clause – Not applicable for CodeCR565**
- 6.2.2.5.44. **Forms - Not applicable for CodeCR565**
- 6.2.2.5.45. **Functions**

Table 48: Functions

Function Name	SaveExamsToMemory
Short Description	Save the split apart exam data (displayed on separate tabs) into memory for later use in saving to a PDF.
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	xFormOutputOKClick	N/A

Function Name	SaveExamsToMemory
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition: N/A
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic
N/A
Modified Logic (Changes are in bold)

```

=====
SaveExamsToMemory
CodeCR565 JRL 06/27/14
This function stores each exam result in memory.

Since the results tabs are created dynamically, cycle
through the tabs to generate the exam text and save
in memory.
=====}
procedure TPNCSTForm.SaveExamsToMemory;
Var
  count, startpos, endpos: integer;
  TabCount, TabCounter, i : Integer;
  richText: TRichEdit;
  tempFolder: array[0..MAX_PATH] of Char;
begin
  If xOutputDesign.Lines.Count = 0 Then
    exit;
  xSampleReportOutput.Visible := false;
  xSampleReportOutput.Lines.Clear;
  xSampleReportOutput.plaintext := false;

  screen.cursor := crHourglass;
  TabCount := xTabControlReport.Tabs.Count;

  if TabCount = 1 then
  begin // if there is only one tab, then there is just one template
    frmMain.NumExamTextForPDFs := 1; // full report is only tab
    SetLength(frmMain.ExamTextForPDFs,1);
  end
  else
  begin
    frmMain.NumExamTextForPDFs := TabCount-1; // Don't include full report tab
    SetLength(frmMain.ExamTextForPDFs, TabCount-1);
  end;

  for TabCounter := 0 to frmMain.NumExamTextForPDFs-1 do
  begin
    startpos := 0;
    count := 0; // Remember that the first tab is for all reports
    Repeat
      Begin
        endpos := xOutputDesign.FindText('// // // !!!@#$FORMENDS$#@!', startpos, Length(xOutputDesign.Text) - startpos, []);
        inc(count);
        If TabCounter+1 = count Then
          Begin
            xRichEditPascalScript.Text := copy(xOutputDesign.text, startpos, endpos - 1);
            CompileAndRunReport;
            count := -1; // Force routine to exit
          End;
          startpos := xOutputDesign.FindText('// // // !!!@#$FORMSTART$#@!', endpos, Length(xOutputDesign.Text) - endpos,
[]);
        End
      End
    until count = 0;
  end;
end

```

```

Until (StartPos = -1) Or (count = -1);

If AuthorName <> xAuthorButton.Caption Then
Begin
  xSampleReportOutput.Lines.Add("");
  xSampleReportOutput.Lines.Add('THIS DOCUMENT WAS ORIGINALLY INITIATED BY: ' + xAuthorButton.Caption);
End;

// take rich text output and change to plain text
richText := TRichEdit.CreateParented(HWND_MESSAGE);
//set MaxLength to the maximum of $7FFFFFFD to allow for large document
//sizes to be combined and converted to plain text
richText.MaxLength := 2147483645;
richText.Clear;
richText.Lines := xSampleReportOutput.Lines;
richText.PlainText := False;
richText.PlainText := True;
frmMain.ExamTextForPDFs[TabCounter] := TMemoryStream.create;
richText.Lines.SaveToStream(frmMain.ExamTextForPDFs[TabCounter]);
xSampleReportOutput.Lines.Clear;
richText.Free;
end;
end;

```

Table 48: Functions

Function Name	xFormOutputOKClick
Short Description	Ok button routine on PNCSMain form.
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	SaveExamsToMemory

Function Name	xFormOutputOKClick
Data Dictionary (DD) References	N/A
Related Protocols	N/A

Function Name	xFormOutputOKClick
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name:N/A Definition: N/A
Output Attribute Name and Definition	Name:N/A Definition:N/A

Current Logic

```

Procedure TPNCSForm.xFormOutputOKClick(Sender: TObject);
Var
  x, xx: integer;
  date1, date2, date3: String;
  foundcptitle: boolean;
  SaveText : TStringList; // CodeCR565 JRL 7/25/14

Begin

  // Make sure the full report is showing
  If xLabelFormOutput.visible true Then
    If xTabControlReport.TabIndex <> 0 Then
      Begin
        xTabControlReport.TabIndex : 0;
        xTabControlReportChange(self);
      End;

    xPanelBaseControl.Visible : True;

    If xButtonFormOutputCancel.Visible False Then
      Begin
        xPanelFormOutput.visible : false;
        xTimerAutosave.enabled : True;
        exit;
      End;

    ShowingReport : False;

    xPanelBaseControl.Visible : True;

    // Check whether to bring up signature info
    If xButtonFormOutputCancel.Visible True Then
      Begin
        TIUSignForm : TTIUSignForm.Create(pncsForm);

        TIUSignForm.ptIENHidden.Caption : xPatientIENS.Caption;

        TIUSignForm.FMReportText.Lines.Clear;
        If xSampleReportOutput.Lines.Count > 0 Then
          For xx : 0 To XSampleReportOutput.Lines.Count 1 Do
            TIUSignForm.FMReportText.Lines.Add(XSampleReportOutput.Lines[xx]);

          {      END OF REPORT      }

          {Get User Name}
          TIUSignForm.FMGets3.IENS : AuthorIEN;
          TIUSignForm.FMGets3.GetAndFill;

          // Tell it which form to use
          TIUSignForm.xEditFormNum.Text : xEditFormNum.Text;

          {Load Titles}
          {Delete off anything inactive and anything not a title}
          TIUSignForm.PNTitles.Items.Clear;
          For x : 0 To xFMPNTitles.Items.Count 1 Do
            Begin
              If Pos(' 11 ', xFMPNTitles.Items[x]) > 0 Then

```

```

        TIUSignForm.PNTitles.Items.Add(Copy(xFMPNTitles.Items[x], 1, pos(' 11 ', xFMPNTitles.Items[x]) 1));
End;
{Screen for selected TITLE(s)}
If pncsForm.Hint <> '' Then
For x : TIUSignForm.PNTitles.Items.Count 1 Downto 0 Do
Begin
If Pos(Uppercase(pncsForm.Hint), Uppercase(TIUSignForm.PNTitles.Items[x])) <> 1 Then
TIUSignForm.PNTitles.Items.Delete(x);
If TIUSignForm.PNTitles.Items.Count > 0 Then
TIUSignForm.PNTitles.ItemIndex : 0;
End;
If TIUSignForm.PNTitles.Items.Count 0 Then
Begin // Reload titles
TIUSignForm.PNTitles.Items.Clear;
For x : 0 To xFMPNTitles.Items.Count 1 Do
Begin
If Pos(' 11 ', xFMPNTitles.Items[x]) > 0 Then
TIUSignForm.PNTitles.Items.Add(Copy(xFMPNTitles.Items[x], 1, pos(' 11 ', xFMPNTitles.Items[x]) 1));
End;
End;

// Screen for C&P Titles if there are any
foundcptitle : false;
If TIUSignForm.PNTitles.Items.Count > 0 Then
For x : 0 To TIUSignForm.PNTitles.Items.Count 1 Do
If Pos('C&P', TIUSignForm.PNTitles.Items[x]) > 0 Then
FoundCPTitle : True;
If TIUSignForm.PNTitles.Items.Count > 0 Then
For x : 0 To TIUSignForm.PNTitles.Items.Count 1 Do
If Pos('PENSION', TIUSignForm.PNTitles.Items[x]) > 0 Then
FoundCPTitle : True;
If TIUSignForm.PNTitles.Items.Count > 0 Then
For x : 0 To TIUSignForm.PNTitles.Items.Count 1 Do
If Pos('COMPENSATION', TIUSignForm.PNTitles.Items[x]) > 0 Then
FoundCPTitle : True;
If TIUSignForm.PNTitles.Items.Count > 0 Then
For x : 0 To TIUSignForm.PNTitles.Items.Count 1 Do
If Pos('COMP AND PEN', TIUSignForm.PNTitles.Items[x]) > 0 Then
FoundCPTitle : True;
If TIUSignForm.PNTitles.Items.Count > 0 Then
For x : 0 To TIUSignForm.PNTitles.Items.Count 1 Do
If Pos('COMP &', TIUSignForm.PNTitles.Items[x]) > 0 Then
FoundCPTitle : True;

If FoundCPTitle true Then
For x : TIUSignForm.PNTitles.Items.Count 1 Downto 0 Do
If ((Pos('C&P', TIUSignForm.PNTitles.Items[x]) 0) And
(Pos('PENSION', TIUSignForm.PNTitles.Items[x]) 0) And
(Pos('COMPENSATION', TIUSignForm.PNTitles.Items[x]) 0) And
(Pos('COMP AND PEN', TIUSignForm.PNTitles.Items[x]) 0) And
(Pos('COMP &', TIUSignForm.PNTitles.Items[x]) 0)) Then
TIUSignForm.PNTitles.Items.Delete(x);

{ Get AMIE Exam Requests}
TIUSignForm.FMExamRequestLister1.PartList.Clear;
TIUSignForm.FMExamRequestLister1.PartList.Add(xPatientIENS.Caption);
TIUSignForm.FMExamRequestListBox.GetList;
// Make sure data is correct for PatientIEN
If TIUSignForm.FMExamRequestListBox.Items.Count > 0 Then
For x : TIUSignForm.FMExamRequestListBox.Items.Count 1 Downto 0 Do
Begin
If Pos(xPatientIENS.Caption + ' ', TIUSignForm.FMExamRequestListBox.Items[x]) <> 1 Then
TIUSignForm.FMExamRequestListBox.Items.Delete(x);
End;
// Reformat data
// Don't show complete or cancelled exams
If TIUSignForm.FMExamRequestListBox.Items.Count > 0 Then
For x : TIUSignForm.FMExamRequestListBox.Items.Count 1 Downto 0 Do
Begin
TIUSignForm.FMExamRequestListBox.Items[x] : Copy(TIUSignForm.FMExamRequestListBox.Items[x], Pos(' ',
TIUSignForm.FMExamRequestListBox.Items[x]) + 4, 255); // Strip patient IEN;
Date1 : Copy(TIUSignForm.FMExamRequestListBox.Items[x], 1, Pos(' ', TIUSignForm.FMExamRequestListBox.Items[x]) 1);
TIUSignForm.FMExamRequestListBox.Items[x] : Copy(TIUSignForm.FMExamRequestListBox.Items[x], Pos(' ',
TIUSignForm.FMExamRequestListBox.Items[x]) + 4, 500);
Date2 : Copy(TIUSignForm.FMExamRequestListBox.Items[x], 1, Pos(' ', TIUSignForm.FMExamRequestListBox.Items[x]) 1);
TIUSignForm.FMExamRequestListBox.Items[x] : Copy(TIUSignForm.FMExamRequestListBox.Items[x], Pos(' ',
TIUSignForm.FMExamRequestListBox.Items[x]) + 4, 500);
Date3 : Copy(TIUSignForm.FMExamRequestListBox.Items[x], 1, 254);
If Date2 <> '' Then
TIUSignForm.FMExamRequestListBox.Items[x] : Copy(FMDateTimeConvert(Date1) + ' ', 1, 20) + ' | ' +
FMDateTimeConvert(Date2)
Else
If Date3 <> '' Then
TIUSignForm.FMExamRequestListBox.Items[x] : Copy(FMDateTimeConvert(Date1) + ' ', 1, 20) + ' | ' +
FMDateTimeConvert(Date3) + ' [EXAM CANCELED]'
Else
TIUSignForm.FMExamRequestListBox.Items[x] : Copy(FMDateTimeConvert(Date1) + ' ', 1, 20) + ' ';
//Fix single digit date and move over
If Copy(TIUSignForm.FMExamRequestListBox.Items[x], 6, 1) ' ' Then
TIUSignForm.FMExamRequestListBox.Items[x] : Copy(TIUSignForm.FMExamRequestListBox.Items[x], 1, 4) + ' ' +
Copy(TIUSignForm.FMExamRequestListBox.Items[x], 5, 15) +
Copy(TIUSignForm.FMExamRequestListBox.Items[x], 21, 99);
If (Date2 <> '') Or (Date3 <> '') Then
TIUSignForm.FMExamRequestListBox.Items.Delete(x);
End;
End;

```

```

(Default to Event Type)
TIUSignForm.ComboBox1.ItemIndex : 3;

TIUSignForm.ButtonAllPreviousAppointmentsClick(Application);

TIUSignForm.ListBox4.Items.Clear; {Inpt Box}
TIUSignForm.ListBox5.Items.Clear; {Hidden Box}

{Use CPRS Visit Loader ORWPT ADMITLST}
frmMain.RPCBroker1.RemoteProcedure : 'ORWPT ADMITLST';
frmMain.RPCBroker1.Param[0].Value : xPatientIENS.Caption;
frmMain.RPCBroker1.Param[0].PType : literal;
frmMain.RPCBrokerCall;
Try
    frmMain.RPCBroker1.Call;
Except
    On EBrokerError Do
        ShowMessageCAPRI('Connection to server for ORWPT ADMITLST could not be established!');
End;
TIUSignForm.ListBox2.Items.Clear; {Hidden Box}
TIUSignForm.ListBox4.Items.Clear; {Inpt Box}
TIUSignForm.ListBox5.Items.Clear; {Hidden Box}
TIUSignForm.ListBox2.Items : frmMain.RPCBroker1.Results;
If TIUSignForm.ListBox2.Items.Count > 0 Then
    For x : 0 To TIUSignForm.ListBox2.Items.Count - 1 Do
        Begin
            TIUSignForm.ListBox4.Items.Add(Copy(TIUSignForm.ListBox2.Items[x], Pos('^', TIUSignForm.ListBox2.Items[x]) + 1, 254));
            TIUSignForm.ListBox4.Items[x] : Copy(TIUSignForm.ListBox4.Items[x], Pos('^', TIUSignForm.ListBox4.Items[x]) + 1, 254);
            TIUSignForm.ListBox4.Items[x] : FMDatetimeConvert(Copy(TIUSignForm.ListBox2.Items[x], 1, Pos('^',
TIUSignForm.ListBox2.Items[x]) - 1) + ' ' + Copy(TIUSignForm.ListBox4.Items[x], 1,
Pos('^', TIUSignForm.ListBox4.Items[x]) - 1);
            TIUSignForm.ListBox5.Items.Add(Piece(TIUSignForm.ListBox2.Items[x], '^', 1) + ';' + Piece(TIUSignForm.ListBox2.Items[x],
'^', 2)); // Get 1st Piece, Strip off V;
        End;
    End;
    TIUSignForm.FMExProvider.Text : Copy(AuthorName, 1, 30);
    TIUSignForm.ComboBoxExamLocation.ItemIndex : 0;
    ShuttingDown : True;
    SignAbortDontKnowWhy : False;
    ContextorChangeMessage : 'You are signing a template. If you continue, CAPRI will drop out of the clinical context.';
    CCOWBreakLink : True;

If TIUSignForm.ShowModal = mrOK Then
Begin
    FreeAndNil(TIUSignForm);
    ShuttingDown : True;
    SignAbortDontKnowWhy : True;
    If ShowingReport = False Then
        xPanelFormOutput.Visible : False;
        SaveName : 'After Signature Screen Activation';
        //pncsForm.x.Button2Click(xButtonCancelYes);
        RecordAudit('Start Save on xFormOutputOKClick', 'Audit');
        BackupAndSave(xButtonCancelYes, xButtonCancelYes, False);
        RecordAudit('End Save on xFormOutputOKClick', 'Audit');
        formstarted : false;
        If FormRunning = true Then
            PNCsForm.PNCsAbortFormRun;
            xStopTimer.Enabled : False;
            xTimerAutoSave.Enabled : False;
            xTimerInfo.Enabled : False;
            pncsForm.visible : false;
            xTimerShutDown.Enabled : True;
            ContextorChangeMessage : '';
            CCOWBreakLink : False;
            exit;
        End
    Else
        Begin
            // The code keeps coming here when it shouldn't, so this flag aborts it
            cursor : crHourglass;
            xPanelBaseControl.Visible : True;
            ShuttingDown : False;
            If ShowingReport = True Then
                exit;
            If SignAbortDontKnowWhy = True Then
                exit;
            If ShowingReport = False Then
                xPanelFormOutput.Visible : False;
                TIUSignForm.Release;
                xButtonFormOutputCancelClick(Application);
                ContextorChangeMessage : '';
                CCOWBreakLink : False;
                exit;
            End;
        End;
    ContextorChangeMessage : '';
    CCOWBreakLink : False;
End;

```


Modified Logic (Changes are in bold)

```

Procedure TPNCSTForm.xFormOutputOKClick(Sender: TObject);
Var
  x, xx: integer;
  date1, date2, date3: String;
  foundcptitle: boolean;
  SaveText : TStringList; // CodeCR565 JRL 7/25/14

Begin
  SaveText := TStringList.Create; // Save Text for TIU Note // CodeCR565 JRL 7/25/14
  SaveText.Assign(xSampleReportOutput.Lines); // CodeCR565 JRL 7/25/14
  SaveExamsToMemory; // CodeCR565 JRL 7/25/14
  xSampleReportOutput.Lines.Assign(SaveText); // CodeCR565 JRL 7/25/14
  FreeAndNil (SaveText); // CodeCR565 JRL 7/25/14

  // Make sure the full report is showing
  If xLabelFormOutput.visible = true Then
    If xTabControlReport.TabIndex <> 0 Then
      Begin
        xTabControlReport.TabIndex := 0;
        xTabControlReport.Change(self);
      End;

  xPanelBaseControl.Visible := True;

  If xButtonFormOutputCancel.Visible = False Then
    Begin
      xPanelFormOutput.visible := false;
      xTimerAutosave.enabled := True;
      exit;
    End;

  ShowingReport := False;

  xPanelBaseControl.Visible := True;

  // Check whether to bring up signature info
  If xButtonFormOutputCancel.Visible = True Then
    Begin
      TIUSignForm := TTIUSignForm.Create(pncsForm);

      TIUSignForm.ptIENHidden.Caption := xPatientIENS.Caption;

      TIUSignForm.FMReportText.Lines.Clear;
      If xSampleReportOutput.Lines.Count > 0 Then
        For xx := 0 To xSampleReportOutput.Lines.Count - 1 Do
          TIUSignForm.FMReportText.Lines.Add(xSampleReportOutput.Lines[xx]);

      {
        END OF REPORT
      }

      {Get User Name}
      TIUSignForm.FMGets3.IENS := AuthorIEN;
      TIUSignForm.FMGets3.GetAndFill;

      // Tell it which form to use
      TIUSignForm.xEditFormNum.Text := xEditFormNum.Text;

      {Load Titles}
      {Delete off anything inactive and anything not a title}
      TIUSignForm.PNTitles.Items.Clear;
      For x := 0 To xFMPNTitles.Items.Count - 1 Do
        Begin
          If Pos(' 11 ', xFMPNTitles.Items[x]) > 0 Then
            TIUSignForm.PNTitles.Items.Add(Copy(xFMPNTitles.Items[x], 1, pos(' 11 ', xFMPNTitles.Items[x]) - 1));
        End;
      {Screen for selected TITLE(s)}
      If pncsForm.Hint <> '' Then
        For x := TIUSignForm.PNTitles.Items.Count - 1 Downto 0 Do
          Begin
            If Pos(Uppercase(pncsForm.Hint), Uppercase(TIUSignForm.PNTitles.Items[x])) <> 1 Then
              TIUSignForm.PNTitles.Items.Delete(x);
            If TIUSignForm.PNTitles.Items.Count > 0 Then
              TIUSignForm.PNTitles.ItemIndex := 0;
            End;
          If TIUSignForm.PNTitles.Items.Count = 0 Then
            Begin // Reload titles

```

```

TIUSignForm.PNTitles.Items.Clear;
For x := 0 To xFMPNTitles.Items.Count - 1 Do
Begin
  If Pos(' 11 ', xFMPNTitles.Items[x]) > 0 Then
    TIUSignForm.PNTitles.Items.Add(Copy(xFMPNTitles.Items[x], 1, pos(' 11 ', xFMPNTitles.Items[x])
1));
  End;
End;

// Screen for C&P Titles if there are any
foundcptitle := false;
If TIUSignForm.PNTitles.Items.Count > 0 Then
  For x := 0 To TIUSignForm.PNTitles.Items.Count - 1 Do
    If Pos('C&P', TIUSignForm.PNTitles.Items[x]) > 0 Then
      FoundCPTitle := True;
  If TIUSignForm.PNTitles.Items.Count > 0 Then
    For x := 0 To TIUSignForm.PNTitles.Items.Count - 1 Do
      If Pos('PENSION', TIUSignForm.PNTitles.Items[x]) > 0 Then
        FoundCPTitle := True;
  If TIUSignForm.PNTitles.Items.Count > 0 Then
    For x := 0 To TIUSignForm.PNTitles.Items.Count - 1 Do
      If Pos('COMPENSATION', TIUSignForm.PNTitles.Items[x]) > 0 Then
        FoundCPTitle := True;
  If TIUSignForm.PNTitles.Items.Count > 0 Then
    For x := 0 To TIUSignForm.PNTitles.Items.Count - 1 Do
      If Pos('COMP AND PEN', TIUSignForm.PNTitles.Items[x]) > 0 Then
        FoundCPTitle := True;
  If TIUSignForm.PNTitles.Items.Count > 0 Then
    For x := 0 To TIUSignForm.PNTitles.Items.Count - 1 Do
      If Pos('COMP &', TIUSignForm.PNTitles.Items[x]) > 0 Then
        FoundCPTitle := True;

If FoundCPTitle = true Then
  For x := TIUSignForm.PNTitles.Items.Count - 1 Downto 0 Do
    If ((Pos('C&P', TIUSignForm.PNTitles.Items[x]) = 0) And
      (Pos('PENSION', TIUSignForm.PNTitles.Items[x]) = 0) And
      (Pos('COMPENSATION', TIUSignForm.PNTitles.Items[x]) = 0) And
      (Pos('COMP AND PEN', TIUSignForm.PNTitles.Items[x]) = 0) And
      (Pos('COMP &', TIUSignForm.PNTitles.Items[x]) = 0)) Then
      TIUSignForm.PNTitles.Items.Delete(x);

{ Get AMIE Exam Requests}
TIUSignForm.FMExamRequestLister1.PartList.Clear;
TIUSignForm.FMExamRequestLister1.PartList.Add(xPatientIENS.Caption);
TIUSignForm.FMExamRequestListBox.GetList;
// Make sure data is correct for PatientIEN
If TIUSignForm.FMExamRequestListBox.Items.Count > 0 Then
  For x := TIUSignForm.FMExamRequestListBox.Items.Count - 1 Downto 0 Do
    Begin
      If Pos(xPatientIENS.Caption + ' ', TIUSignForm.FMExamRequestListBox.Items[x]) <> 1 Then
        TIUSignForm.FMExamRequestListBox.Items.Delete(x);
    End;
  // Reformat data
  // Don't show complete or cancelled exams
  If TIUSignForm.FMExamRequestListBox.Items.Count > 0 Then
    For x := TIUSignForm.FMExamRequestListBox.Items.Count - 1 Downto 0 Do
      Begin
        TIUSignForm.FMExamRequestListBox.Items[x] := Copy(TIUSignForm.FMExamRequestListBox.Items[x], Pos(' ',
TIUSignForm.FMExamRequestListBox.Items[x]) + 4, 255); // Strip patient IEN;
        Date1 := Copy(TIUSignForm.FMExamRequestListBox.Items[x], 1, Pos(' ',
TIUSignForm.FMExamRequestListBox.Items[x]) - 1);
        TIUSignForm.FMExamRequestListBox.Items[x] := Copy(TIUSignForm.FMExamRequestListBox.Items[x], Pos(' ',
TIUSignForm.FMExamRequestListBox.Items[x]) + 4, 500);
        Date2 := Copy(TIUSignForm.FMExamRequestListBox.Items[x], 1, Pos(' ',
TIUSignForm.FMExamRequestListBox.Items[x]) - 1);
        TIUSignForm.FMExamRequestListBox.Items[x] := Copy(TIUSignForm.FMExamRequestListBox.Items[x], Pos(' ',
TIUSignForm.FMExamRequestListBox.Items[x]) + 4, 500);
        Date3 := Copy(TIUSignForm.FMExamRequestListBox.Items[x], 1, 254);
        If Date2 <> '' Then
          TIUSignForm.FMExamRequestListBox.Items[x] := Copy(FMDateTimeConvert(Date1) + '
1, 20) + ' | ' + FMDateTimeConvert(Date2)
        Else
          If Date3 <> '' Then
            TIUSignForm.FMExamRequestListBox.Items[x] := Copy(FMDateTimeConvert(Date1) + '
', 1, 20) + ' | ' + FMDateTimeConvert(Date3) + ' [EXAM CANCELED]'
          Else
            TIUSignForm.FMExamRequestListBox.Items[x] := Copy(FMDateTimeConvert(Date1) + '
', 1, 20) + ' ';
          //Fix single digit date and move over
          If Copy(TIUSignForm.FMExamRequestListBox.Items[x], 6, 1) = ',' Then

```

```

        TIUSignForm.FMExamRequestListbox.Items[x] := Copy(TIUSignForm.FMExamRequestListbox.Items[x], 1, 4) +
' ' + Copy(TIUSignForm.FMExamRequestListbox.Items[x], 5, 15) +
        Copy(TIUSignForm.FMExamRequestListbox.Items[x], 21, 99);
        If (Date2 <> '') Or (Date3 <> '') Then
            TIUSignForm.FMExamRequestListbox.Items.Delete(x);
        End;

{Default to Event Type}
TIUSignForm.ComboBox1.ItemIndex := 3;

TIUSignForm.ButtonAllPreviousAppointmentsClick(Application);

TIUSignForm.ListBox4.Items.Clear; {Inpt Box}
TIUSignForm.ListBox5.Items.Clear; {Hidden Box}

{Use CPRS Visit Loader ORWPT ADMITLST}
frmMain.RPCBroker1.RemoteProcedure := 'ORWPT ADMITLST';
frmMain.RPCBroker1.Param[0].Value := xPatientIENS.Caption;
frmMain.RPCBroker1.Param[0].PType := literal;
frmMain.RPCBrokerCall;
Try
    frmMain.RPCBroker1.Call;
Except
    On EBrokerError Do
        ShowMessageCAPRI('Connection to server for ORWPT ADMITLST could not be established!');
End;
TIUSignForm.ListBox2.Items.Clear; {Hidden Box}
TIUSignForm.ListBox4.Items.Clear; {Inpt Box}
TIUSignForm.ListBox5.Items.Clear; {Hidden Box}
TIUSignForm.ListBox2.Items := frmMain.RPCBroker1.Results;
If TIUSignForm.ListBox2.Items.Count > 0 Then
    For x := 0 To TIUSignForm.ListBox2.Items.Count - 1 Do
        Begin
            TIUSignForm.ListBox4.Items.Add(Copy(TIUSignForm.ListBox2.Items[x], Pos('^',
TIUSignForm.ListBox2.Items[x]) + 1, 254));
            TIUSignForm.ListBox4.Items[x] := Copy(TIUSignForm.ListBox4.Items[x], Pos('^',
TIUSignForm.ListBox4.Items[x]) + 1, 254);
            TIUSignForm.ListBox4.Items[x] := FMDateTimeConvert(Copy(TIUSignForm.ListBox2.Items[x], 1, Pos('^',
TIUSignForm.ListBox2.Items[x]) - 1)) + ' ' + Copy(TIUSignForm.ListBox4.Items[x], 1,
            Pos('^', TIUSignForm.ListBox4.Items[x]) - 1);
            TIUSignForm.ListBox5.Items.Add(Piece(TIUSignForm.ListBox2.Items[x], '^', 1) + ';' +
Piece(TIUSignForm.ListBox2.Items[x], '^', 2)); // Get 1st Piece, Strip off V;
        End;
        TIUSignForm.FMExProvider.Text := Copy(AuthorName, 1, 30);
        TIUSignForm.ComboBoxExamLocation.ItemIndex := 0;
        ShuttingDown := True;
        SignAbortDontKnowWhy := False;
        ContextorChangeMessage := 'You are signing a template. If you continue, CAPRI will drop out of the
clinical context.';
        CCOWBreakLink := True;

If TIUSignForm.ShowModal = mrOK Then
    Begin
        FreeAndNil(TIUSignForm);
        ShuttingDown := True;
        SignAbortDontKnowWhy := True;
        If ShowingReport = False Then
            xPanelFormOutput.Visible := False;
        SaveName := 'After Signature Screen Activation';
        //pncsForm.x Button2Click(xButtonCancelYes);
        RecordAudit('Start Save on xFormOutputOKClick', 'Audit');
        BackupAndSave(xButtonCancelYes, xButtonCancelYes, False);
        RecordAudit('End Save on xFormOutputOKClick', 'Audit');
        formstarted := false;
        If FormRunning = true Then
            PNCSForm.PNCSAbortFormRun;
            xStopTimer.Enabled := False;
            xTimerAutoSave.Enabled := False;
            xTimerInfo.Enabled := False;
            pncsForm.visible := false;
            xTimerShutDown.Enabled := True;
            ContextorChangeMessage := '';
            CCOWBreakLink := False;
        exit;
    End
Else
    Begin
        // The code keeps coming here when it shouldn't, so this flag aborts it
        cursor := crHourglass;
        xPanelBaseControl.Visible := True;

```

```

ShuttingDown := False;
If ShowingReport = True Then
    exit;
If SignAbortDontKnowWhy = True Then
    exit;
If ShowingReport = False Then
    xPanelFormOutput.Visible := False;
TIUSignForm.Release;
xButtonFormOutputCancelClick(Application);
ContextorChangeMessage := '';
CCOWBreakLink := False;
exit;
End;
End;
ContextorChangeMessage := '';
CCOWBreakLink := False;
End;

```

6.2.2.5.46. GUI

Table 36: GUI

Unit Name	Description
TIUSign.pas	Alter the units mentioned to take the currently “merged” PDF sent to both Virtual VA and VLER DAS and split it out into a single PDF per exam.

6.2.2.5.47. GUI Classes - Not applicable for CodeCR565

6.2.2.5.48. GUI Classes - - Not applicable for CodeCR565

6.2.2.5.49. Current Form - Not applicable for CodeCR565

6.2.2.5.50. Modified Form - Not applicable for CodeCR565

6.2.2.5.51. Components on Form - Not applicable for CodeCR565

6.2.2.5.52. Events

Table 40: Events

Name	Type	Description
ButtonOK2Click	Procedure	Modify routine to send the PDFs saved in memory and store them in the memory used for XML PDF attachments.

- 6.2.2.5.53. **Methods – Not applicable for CodeCR565**
- 6.2.2.5.54. **Special References - Not applicable for CodeCR565**
- 6.2.2.5.55. **Class Events - Not applicable for CodeCR565**
- 6.2.2.5.56. **Class Methods - Not applicable for CodeCR565**
- 6.2.2.5.57. **Class Properties - Not applicable for CodeCR565**
- 6.2.2.5.58. **Uses Clause – Not applicable for CodeCR565**
- 6.2.2.5.59. **Forms - Not applicable for CodeCR565**
- 6.2.2.5.60. **Functions**

Table 48: Functions

Function Name	ButtonOK2Click
Short Description	Modify routine to send the PDFs saved in memory and store them in the memory used for XML PDF attachments.
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	N/A

Function Name	ButtonOK2Click
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition: N/A
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic

N/A

```
procedure TTIUSignForm.ButtonOK2Click(Sender: TObject);
var
  Appointment: String;
  TitleIEN: String;
  ErrMsg: String;
  CheckTIUNotes : Boolean;
  SendExamToVVA : Boolean;           // CodeCR423 JRL 3/8/13
  PopulatingNameSSN : Boolean;       // CodeCR423 JRL 3/8/13
  I : Integer;                      // CodeCR471 JRL 4/30/13
  ExamIENS : TStringList;           // CodeCR471 JRL 4/30/13
  RequestIEN : String;              // CodeCR471 JRL 5/08/13
  buffer : string;                  // CodeCR471 JRL 5/08/13
  SelectedPatientName, SavePatientName : String; // JRL 5/3/13
  SelectedPatientSSN, SavePatientSSN : String; // JRL 5/3/13
  CosignYes_CnPExamNO : Boolean;    // CodeCR459 JRL 7/18/13
  CnPExam : Boolean;                // CodeCR459 JRL 7/22/13
  selectedPatient: TPatient;

  {cosignature variables needed for vler das #186}
  isCosignatureRequired : Boolean;
  cosigTiuDocumentIEN: String;
  cosigCapriTemplateIEN: String;
  TransmissionErrors : Boolean;    //CodeCR562 JRL 7/17/14

begin
  { is a cosignature required? }
  isCosignatureRequired : Trim(FMedit16.Text) <> '';

  if Self.Owner is TfrmTIUCosign then
    cosigTiuDocumentIEN : (Self.Owner as TfrmTIUCosign).TiuDocumentIEN;

  { identify which exam will be the parent exam ien, addresses an edge case bug }
  for i : 0 To lstExamsRequested.Items.Count - 1 do
  begin
    if lstExamsRequested.Selected[i] = true then
    begin
      parentExamIen : Piece(lstExamsRequested.Items[i], '^', 2);
      ParentExamTitle : TrimRight(Piece(lstExamsRequested.Items[i], '^', 1)); // JRL 09/27/13 C&P Exam Title
      break;
    end;
  end;

  // Is the question "Is this a C&P exam" being displayed?
  if cmbxCPRqst.Visible then //CodeCR459 JRL 7/22/13
    CnPExam : SameText(UpperCase(cmbxCPRqst.Text), 'YES') //CodeCR459 JRL 5/21/13
  else //CodeCR459 JRL 7/22/13
    CnPExam : TRUE; //CodeCR459 JRL 7/22/13

  // If the user chooses NO to "is this a C&P exam" and there is a cosignature,
  // handle the same way as "NO" without a cosignature, i.e. mark the exam as
  // complete.
  if (CnPExam = FALSE) and //CodeCR459 JRL 7/22/13
    (cosignerButton.Caption <> '') then //CodeCR459 JRL 7/18/13
    CosignYes_CnPExamNO : TRUE //CodeCR459 JRL 7/18/13
  else //CodeCR459 JRL 7/18/13
    CosignYes_CnPExamNO : FALSE; //CodeCR459 JRL 7/18/13

  ComboBoxExamLocation.ItemIndex : 0;

  // Set to release report in AMIE by default.
  ComboBoxReleaseReport.ItemIndex : 0;

  CosigMode : False;
  if CoSigTIUNoteIEN <> '' then
  begin
    TIUNoteIEN : CoSigTIUNoteIEN;
    CosigMode : True;
  end;

  //Validate user input fields Short circuit on failure
  if NOT IsUserInputValid(CosigMode) then
    Exit;

  // wait until validation is done. Then check if the condition of a Non C&P
  // Exam is being signed that requires a cosignature. If so, delete the
  // cosignature information and send the TIU note only. Removing the cosignature
  // information keeps the cosigner from getting an "alert" when they login to
  // CAPRI. The cosignature process is handled outside of CAPRI for non C&P
  // exams so we don't care who the cosigner was supposed to be. Bottom line is
  // sacrificing storing this data to avoid the alert.
  if CosignYes_CnPExamNO = TRUE then //CodeCR459 JRL 7/25/13
  begin //CodeCR459 JRL 7/25/13
    cosignerButton.Caption : ''; //CodeCR459 JRL 7/25/13
    FMedit16.Text : ''; // Clear out saved IEN //CodeCR459 JRL 7/25/13
  end; //CodeCR459 JRL 7/25/13

  //Are any exams pending Short circuit if NO
  if CnPExam then //CodeCR459 JRL 5/21/13
  begin
```

```

        if FMExamRequestListbox.Items.Count = 0 then
        begin
            ShowMessageCAPRI('There are no AMIE exam requests needing results. This document must be linked to an open request before
it can be uploaded.');
```

ModalResult : mrCancel;

Exit;

end;

end; { End If Template was for C&P Request }

```

        if CosigMode = False then
        begin {Start NOT CosigMode}
            { Swap date and clinic location }
            VisitIDIEN : ''; // clear any previous visit IENS //CodeCR499 JRL 6/27/13
            if Listbox4.ItemIndex > 1 then
                VisitIDIEN : Piece(Listbox5.Items[Listbox4.ItemIndex], ',', 2) + ',';
                + Piece(Listbox5.Items[Listbox4.ItemIndex], ',', 1) + ';H'
            else if Listbox1.ItemIndex > 1 then
            begin
                Appointment : FindMatchingApptData;
                VisitIDIEN : Piece(Appointment, '^', 2) + ',';
                + Piece(Appointment, '^', 1) + ';A';
            end;

            { Create and Populate TIU Record }
            if TIUNoteIEN = '' then
            begin
                TitleIEN : GetTIUTitleIEN;
                TIUNoteIEN : CreateTIURecord(VisitIDIEN, TitleIEN);

                if Piece(TIUNoteIEN, '^', 1) = '0' then
                begin
                    ShowMessageCAPRI('Could not create TIU Record. Cannot Continue. ' + Piece(TIUNoteIEN, '^', 2));
                    ModalResult : mrCancel;
                    Exit;
                end;

                // Populate TIU record text
                ErrMsg : '';
                if PNCSForm.xSampleReportOutput.Lines.Count > 0 then
                    SetText(ErrMsg, PNCSForm.xSampleReportOutput.Lines, StrToInt(TIUNoteIEN), 0);

                if ErrMsg <> '' then
                    ShowMessageCAPRI(ErrMsg);
            end;

            PNCSForm.xPanelSave.Visible : False;
            PNCSForm.xGauge1.Progress : 0;
            TIUSignForm.Visible : True;

            { Validate TIU Text transfer }
            if NOT IsTIUTextValid(TIUNoteIEN) then
            begin
                ModalResult : mrCancel;
                Exit;
            end;

            { Sign TIU Record and if failed, then try to delete TIU }
            if SignTIURecord(TIUNoteIEN, Edit3.Text) then
            begin
                //Update status and TIU Document # in CAPRI and AMIE files
                PNCSForm.xFMEdit10.Text : TIUNoteIEN; // Set TIU Document number in CAPRI template file
                // Set Review Status
                if cosignerButton.Caption <> '' then
                    pnCSForm.xFMEditReviewStatus.Text : 'U'
                else
                    pnCSForm.xFMEditReviewStatus.Text : 'C';
                if CosignYes_CnPExamNO then // CodeCR459 JRL 7/22/13 If cosigning a non c&p exam, then set
                    pnCSForm.xFMEditReviewStatus.Text : 'C'; // CodeCR459 JRL 7/22/13 the status to complete.
                pnCSForm.xFMEdit8.Text : 'NOW'; {Set date/time of sig to lock note in IPR file}
            end
            else
            begin
                DeleteTIURecord(TIUNoteIEN, '');
                ModalResult : mrCancel;
                Exit;
            end;
        end {End NOT CosigMode}
    else
    begin {Start CosigMode}
        if SignTIURecord(TIUNoteIEN, Edit3.Text) then
        begin
            if frmTIUCosign.addendumText.Lines.Count > 0 then
                frmTIUCosign.SaveAddendum;
            end
            else
                Exit;
        end;
    end; {End CosigMode}

    // Get Episode Date/Time
    anuApptPointer : ''; //CodeCR499 JRL 6/27/13
    if not CheckBoxOpinion.Checked then //CodeCR499 JRL 6/27/13
        anuApptPointer : GetApptPointer(TIUNoteIEN);

    if CnPExam then //CodeCR459 JRL 7/22/13
    begin //Added if statement and encapsulated

```

```

//existing code

// Set AMIE with TIU text
if labelCosigner.enabled false then
    SetAMIEResults(TIUNoteIEN, VisitIDIEN, anuApptPointer);

//Store linked AMIE exams in CAPRI Template file
SetExamInTemplate(coSigMode);

if labelCosigner.enabled false then
begin
    // Send out email for completed exam(s)
    try
        ExamIENS : TStringList.Create;
        ExamIENS.Clear;
        for I : 0 To lstExamsRequested.Items.Count - 1 do
            begin
                if lstExamsRequested.Selected[I] true then
                    begin
                        buffer : Piece(lstExamsRequested.Items[I], '^', 2);
                        ExamIENS.Add(buffer);
                    end;
                end;
            end;
        RequestIEN : FMExamRequestListBox.GetSelectedRecord.IEN;
        if NOT SendNotificationTwo(ExamIENS,
            AuthorIEN,
            RequestIEN) then
            ShowMessageCAPRI('There was a problem sending a completion email notification. Please notify IRM!');
    finally
        FreeAndNil(ExamIENS);
    end;

    FMExamRequestListBoxClick(Application);
    // Be sure that nothing is still open
    if IsExamOKtoRelease then
        ReleaseExamToRO; // Release the results to the RO

end;
end; { End If Template was for C&P Request }

TIUSignForm.Visible : False;

if CosigMode True then
    // Find template that matches this document in CAPRI and change
    // status to complete.
    SetMatchingTemplateComplete(TIUNoteIEN)
else if isCosignatureRequired True then
begin
    frmMain.RPCBroker1.Results.Clear;
    frmMain.RPCBroker1.Param.Clear;
    frmMain.RPCBroker1.RemoteProcedure : 'DVBA CAPRI EXAM LINK TIU';
    frmMain.RPCBroker1.Param[0].Value : Piece(PNCSForm.xFMedit2.IENS, ',', 1);
    frmMain.RPCBroker1.Param[0].PType : literal;
    frmMain.RPCBroker1.Param[1].Value : TIUNoteIEN;
    frmMain.RPCBroker1.Param[1].PType : literal;
    frmMain.RPCBroker1.Call;
    frmMain.RPCBroker1.Call;
    if frmMain.RPCBroker1.Results.Count > 0 then
        if frmMain.RPCBroker1.Results[0] <> '1' then
            MessageDlgCAPRI('Error: Unable to link exam to progress note.', mtError, [mbOK], 0);
end;

ModalResult : mrOk; //rra 917335 flag 396.7 for update when returns to TPNCForm.xFormOutputOKClick

{ Send exam to VVA
    * Do not send an exam to VVA unless it is an C&P Exam.
    * Do not send exam if a cosignature is needed (CosignerButton.Caption '')
    * CosigMode FALSE, then always send this exam to VVA
    * CosigMode TRUE means this exam required a signature and the cosigner
      is reviewing the exam.
    * If a cosigner is signing the exam from the popup alert, or if an exam is
      being accessed from the Tools | My Unsigned Exams menu option, this means
      the patient may not have been selected yet and the global variables
      (name/ssn) are not populated. If these variables are blank, then get
      the IEN for the patient stored in ptIENHidden.Caption, retrieve the
      name and ssn, and populate the global variables PatientName and
      PatientSSN so these values exist on the main form when the document is
      sent to VVA. Otherwise, the transmission to VVA will error. Then reset
      the variables so blank so nothing is left behind from the transmit.

    Original SendToVVA code:
    Update SendToVVA code handling cosigned exams
    Updated to handle sending to the wrong VVA folder
    because if the same issue as the cosigned exams
    coming from the Alert Popups or the Unsigned Exams
    menu item
    Updated to only send C&P Exams to VVA

    CodeCR423 JRL 11/13/12
    CodeCR423 JRL 03/08/12
    JRL 05/03/13
    JRL 07/22/13

}

if CnPExam then
begin
    PopulatingNameSSN : FALSE;
    SendExamToVVA : FALSE;

    SavePatientName : PatientName;
    SavePatientSSN : PatientSSN;

```



```

selectedPatient : TPatient.Create;

FMGetsVVAPatientInfo.FieldNumbers.Clear;
FMGetsVVAPatientInfo.FieldNumbers.Add('01'); // name
FMGetsVVAPatientInfo.FieldNumbers.Add('09'); // ssn
FMGetsVVAPatientInfo.FieldNumbers.Add('02'); // gender
FMGetsVVAPatientInfo.FieldNumbers.Add('03'); // date of birth format ie, 12/24/1929
FMGetsVVAPatientInfo.FieldNumbers.Add('991.01'); // icn
FMGetsVVAPatientInfo.FieldNumbers.Add('313'); // claim number
FMGetsVVAPatientInfo.FileName : '2';
FMGetsVVAPatientInfo.IENS : ptIENHidden.Caption;
FMGetsVVAPatientInfo.GetData;
if FMGetsVVAPatientInfo.Results.Count > 0 then
begin
    SelectedPatientName : FMGetsVVAPatientInfo.GetField('01').FMDExternal;
    SelectedPatientSSN : FMGetsVVAPatientInfo.GetField('09').FMDExternal;

//    selectedPatient.FirstName : Piece(SelectedPatientName, ',', 1);
//    selectedPatient.LastName : Piece(SelectedPatientName, ',', 1);
//    selectedPatient.MiddleName : Piece(Piece(SelectedPatientName, ',', 2), ' ', 2);
//    selectedPatient.LastName : Piece(Piece(SelectedPatientName, ',', 2), ' ', 1);
//    selectedPatient.FirstName : Piece(Piece(SelectedPatientName, ',', 2), ' ', 1);

    selectedPatient.Gender : FMGetsVVAPatientInfo.GetField('02').FMDInternal;
    selectedPatient.Birthdate : FMGetsVVAPatientInfo.GetField('03').FMDExternal;
    selectedPatient.Birthdate : Piece(selectedPatient.Birthdate, '/', 3) + ' ' +
        Piece(selectedPatient.Birthdate, '/', 1) + ' ' + Piece(selectedPatient.Birthdate, '/', 2);
    selectedPatient.IntegrationControlNumber : FMGetsVVAPatientInfo.GetField('991.01').FMDExternal;
    selectedPatient.SocialSecurityNumber : SelectedPatientSSN;
    selectedPatient.ClaimNumber : FMGetsVVAPatientInfo.GetField('313').FMDExternal;
end
else
begin
    SelectedPatientName : '';
    SelectedPatientSSN : '';
end;

if (PatientName = '') and (PatientSSN = '') and
    (SelectedPatientName = '') and (SelectedPatientSSN = '') then
begin
    SendExamToVVA : FALSE;
    ShowMessageCapri('This exam must be manually transmitted to Virtual VA');
end
else if (PatientName = SelectedPatientName) and (PatientSSN = SelectedPatientSSN) then
begin
    PopulatingNameSSN : FALSE;
    SendExamToVVA : TRUE;
end
else if (PatientName = '') or (PatientSSN = '') then
begin
    PopulatingNameSSN : TRUE;
    PatientName : SelectedPatientName;
    PatientSSN : SelectedPatientSSN;
    SendExamToVVA : TRUE;
end
else if (SelectedPatientName <> PatientName) and (SelectedPatientSSN <> PatientSSN) then
begin
    PopulatingNameSSN : TRUE;
    PatientName : SelectedPatientName;
    PatientSSN : SelectedPatientSSN;
    SendExamToVVA : TRUE;
end
else
begin
    SendExamToVVA : FALSE;
    ShowMessageCapri('This exam must be manually transmitted to Virtual VA');
end;

if CosignerButton.Caption <> '' then
    SendExamToVVA : FALSE;

if SendExamToVVA then
begin
    // Get Report data to automatically send info to the VVA
    TIUtoVVAProgressNotes : TStringList.Create;
    try
        CheckTIUNotes : GetTIUText(TIUNoteIEN, TIUtoVVAProgressNotes);
        if CheckTIUNotes then // Notes returned
            frmMain.actFileTransmitVirtualVAExecute(Self);
        finally
            FreeAndNil(TIUtoVVAProgressNotes);
        end;
    end;

{ if we're sending to VVA then we save to 1) VLER DAS and 2) VistA }
if SendExamToVVA and (isCosignatureRequired = False) then
begin
    if CoSigMode = True then
    begin
        LoadDBQForRendering(cosigTiuDocumentIEN);
        frmMain.NumExamTextForPDFs : pncsForm.Exams.Count;
        SetLength(frmMain.ExamTextForPDFs, frmMain.NumExamTextForPDFs);
        for i : 0 to pncsForm.Exams.Count - 1 do
            begin
                frmMain.ExamTextForPDFs[i] : TMemoryStream.create;
            end;
        end;
    end;
end;

```

```

        FinalReport.SaveToStream(frmMain.ExamTextForPDFs[i]);
    end;
end;
// Created in Virtual VA code in case it is needed for cosignature text.
FreeAndNil(FinalReport); // CodeCR565 JRL 7/20/14

if pncsForm Nil then
    MessageDlg('DBQ is not loaded; will NOT transmit to VLER DAS', mtError, [mbOk], 0)
else
    begin
//      SendDbqsToVlerAndVista(selectedPatient);
        for i : 1 to pncsForm.Exams.Count do // Setup ReTransmit array // CodeCR562 JRL 7/17/14
            ExamSent[i] : FALSE; // CodeCR562 JRL 7/17/14
        SendDbqsToVlerAndVista(selectedPatient,FALSE); // Send first time // CodeCR562 JRL 7/17/14
        TransmissionErrors : FALSE; // CodeCR562 JRL 7/17/14
        for i : 1 to pncsForm.Exams.Count do // CodeCR562 JRL 7/17/14
            begin // CodeCR562 JRL 7/17/14
                if ExamSent[i] FALSE then // CodeCR562 JRL 7/17/14
                    TransmissionErrors : TRUE; // CodeCR562 JRL 7/17/14
                end; // CodeCR562 JRL 7/17/14
                if TransmissionErrors then // CodeCR562 JRL 7/17/14
                    begin // CodeCR562 JRL 7/17/14
                        if MessageDlgCAPRI('Transmission to VLER failed, do you want to retransmit?',mtConfirmation,[mbYes,mbNo],0) mrYes
then
                            SendDbqsToVlerAndVista(selectedPatient,TRUE); // CodeCR562 JRL 7/17/14
                        end; // else // CodeCR562 JRL 7/17/14
                        // free memory for stored PDFs here // CodeCR562 JRL 7/18/14 moved code to accommodate retransmission //
CodeCR565 JRL 7/15/14
                        for i : 0 to frmMain.NumExamTextForPDFs - 1 do // CodeCR562 JRL 7/18/14 // CodeCR565 JRL 7/15/14
                            FreeAndNil(frmMain.ExamTextForPDFs[i]); // CodeCR562 JRL 7/18/14 // CodeCR565 JRL 7/15/14
                        end; // CodeCR562 JRL 7/17/14
                        if CoSigMode True then
                            begin
                                UnloadDBQForRendering;
                            end;
                        end;
                    end;

                    if PopulatingNameSSN then // if we populated the global variables
                        begin // to send the exam to VVA, restore
                            PatientName : SavePatientName; // them now to avoid any other issues
                            PatientSSN : SavePatientSSN; // in CAPRI
                        end;
                        FreeAndNil(SelectedPatient);
                    end; // if CnPExam

                    ModalResult : mrOk;
                end;
            end;
        end;
    end;
end;

```

Modified Logic (Changes are in bold)

```

procedure TTIUSignForm.ButtonOK2Click(Sender: TObject);
var
    Appointment: String;
    TitleIEN: String;
    ErrMsg: String;
    CheckTIUNotes : Boolean;
    SendExamToVVA : Boolean; // CodeCR423 JRL 3/8/13
    PopulatingNameSSN : Boolean; // CodeCR423 JRL 3/8/13
    I : Integer; // CodeCR471 JRL 4/30/13
    ExamIENS : TStrings; // CodeCR471 JRL 4/30/13
    RequestIEN : String; // CodeCR471 JRL 5/08/13
    buffer : string; // CodeCR471 JRL 5/08/13
    SelectedPatientName, SavePatientName : String; // JRL 5/3/13
    SelectedPatientSSN, SavePatientSSN : String; // JRL 5/3/13
    CosignYes_CnPExamNO : Boolean; // CodeCR459 JRL 7/18/13
    CnPExam : Boolean; // CodeCR459 JRL 7/22/13
    selectedPatient: TPatient;

    {cosignature variables needed for vler das #186}
    isCosignatureRequired : Boolean;
    cosigTiuDocumentIEN: String;
    cosigCapriTemplateIEN: String;
    TransmissionErrors : Boolean; //CodeCR562 JRL 7/17/14

begin
    { is a cosignature required? }
    isCosignatureRequired : Trim(FMEdit16.Text) <> '';

    if Self.Owner is TfrmTIUCosign then
        cosigTiuDocumentIEN : (Self.Owner as TfrmTIUCosign).TiuDocumentIEN;

    { identify which exam will be the parent exam ien, addresses an edge case bug }
    for i : 0 To lstExamsRequested.Items.Count - 1 do
        begin
            if lstExamsRequested.Selected[i] true then
                begin
                    parentExamIEN : Piece(lstExamsRequested.Items[i], '^', 2);
                    ParentExamTitle : TrimRight(Piece(lstExamsRequested.Items[i], '^', 1)); // JRL 09/27/13 C&P Exam Title
                    break;
                end;
            end;
        end;
    end;
end;

```

```

// Is the question "Is this a C&P exam" being displayed?
if cmbxCPRqst.Visible then                                     //CodeCR459 JRL 7/22/13
    CnPExam : SameText(UpperCase(cmbxCPRqst.Text), 'YES')      //CodeCR459 JRL 5/21/13
else                                                            //CodeCR459 JRL 7/22/13
    CnPExam : TRUE;                                           //CodeCR459 JRL 7/22/13

// If the user chooses NO to "is this a C&P exam" and there is a cosignature,
// handle the same way as "NO" without a cosignature, i.e. mark the exam as
// complete.
if (CnPExam FALSE) and                                       //CodeCR459 JRL 7/22/13
    (cosignerButton.Caption <> '') then                       //CodeCR459 JRL 7/18/13
    CosignYes_CnPExamNO : TRUE                               //CodeCR459 JRL 7/18/13
else                                                            //CodeCR459 JRL 7/18/13
    CosignYes_CnPExamNO : FALSE;                             //CodeCR459 JRL 7/18/13

ComboBoxExamLocation.ItemIndex : 0;

// Set to release report in AMIE by default.
ComboBoxReleaseReport.ItemIndex : 0;

CosigMode : False;
if CoSigTIUNoteIEN <> '' then
begin
    TIUNoteIEN : CoSigTIUNoteIEN;
    CosigMode : True;
end;

//Validate user input fields Short circuit on failure
if NOT IsUserInputValid(CosigMode) then
    Exit;

// wait until validation is done. Then check if the condition of a Non C&P
// Exam is being signed that requires a cosignature. If so, delete the
// cosignature information and send the TIU note only. Removing the cosignature
// information keeps the cosigner from getting an "alert" when they login to
// CAPRI. The cosignature process is handled outside of CAPRI for non C&P
// exams so we don't care who the cosigner was supposed to be. Bottom line is
// sacrificing storing this data to avoid the alert.
if CosignYes_CnPExamNO TRUE then                               //CodeCR459 JRL 7/25/13
begin                                                            //CodeCR459 JRL 7/25/13
    cosignerButton.Caption : '';                               //CodeCR459 JRL 7/25/13
    FMEdit16.Text : ''; // Clear out saved IEN                 //CodeCR459 JRL 7/25/13
end;                                                            //CodeCR459 JRL 7/25/13

//Are any exams pending Short circuit if NO
if CnPExam then //CodeCR459 JRL 5/21/13
begin
    if FMEExamRequestListbox.Items.Count = 0 then
    begin
        ShowMessageCAPRI('There are no AMIE exam requests needing results. This document must be linked to an open request before
it can be uploaded.');
```

```

{          Validate TIU Text transfer          }
if NOT IsTIUTextValid(TIUNoteIEN) then
begin
    ModalResult : mrCancel;
    Exit;
end;

{          Sign TIU Record and if failed, then try to delete TIU          }
if SignTIURecord(TIUNoteIEN, Edit3.Text) then
begin
    //Update status and TIU Document # in CAPRI and AMIE files
    PNCSForm.xFMEdit10.Text : TIUNoteIEN; // Set TIU Document number in CAPRI template file
    // Set Review Status
    if cosignerButton.Caption <> '' then
        pncsForm.xFMEditReviewStatus.Text : 'U'
    else
        pncsForm.xFMEditReviewStatus.Text : 'C';
    if CosignYes_CnPExamNO then // CodeCR459 JRL 7/22/13 If cosigning a non c&p exam, then set
        pncsForm.xFMEditReviewStatus.Text : 'C'; // CodeCR459 JRL 7/22/13 the status to complete.
    pncsForm.xFMEdit8.Text : 'NOW'; {Set date/time of sig to lock note in IPR file}
end
else
begin
    DeleteTIURecord(TIUNoteIEN, '');
    ModalResult : mrCancel;
    Exit;
end;
end {End NOT CosigMode}
else
begin {Start CosigMode}
    if SignTIURecord(TIUNoteIEN, Edit3.Text) then
    begin
        if frmTIUCosign.addendumText.Lines.Count > 0 then
            frmTIUCosign.SaveAddendum;
        end
    else
        Exit;
    end; {End CosigMode}

    // Get Epsiode Date/Time
    anuApptPointer : ''; //CodeCR499 JRL 6/27/13
    if not CheckBoxOpinion.Checked then //CodeCR499 JRL 6/27/13
        anuApptPointer : GetApptPointer(TIUNoteIEN);

    if CnPExam then //CodeCR459 JRL 7/22/13
    begin //Added if statement and encapsulated
        //existing code

        // Set AMIE with TIU text
        if labelCosigner.enabled false then
            SetAMIEResults(TIUNoteIEN, VisitIDIEN, anuApptPointer);

        //Store linked AMIE exams in CAPRI Template file
        SetExamInTemplate(coSigMode);

        if labelCosigner.enabled false then
        begin
            // Send out email for completed exam(s) // Added email send routine
            try // from try/finally inclusive
                ExamIENs : TStringList.Create; // CodeCR471 JRL 5/1/13
                ExamIENs.Clear; // Relocated code JRL 6/25/13
                for I : 0 To lstExamsRequested.Items.Count - 1 do //
                    begin //
                        if lstExamsRequested.Selected[I] true then // Get selected exam IEN
                            begin // and add to exam list
                                buffer : Piece(lstExamsRequested.Items[I], '^', 2); //
                                ExamIENs.Add(buffer); //
                            end; //
                        end; //
                    end; //
                RequestIEN : FMEExamRequestListBox.GetSelectedRecord.IEN; //
                if NOT SendNotificationTwo(ExamIENs, //
                    AuthorIEN, //
                    RequestIEN) then //
                    ShowMessageCAPRI('There was a problem sending a completion email notification. Please notify IRM!'); //
                finally //
                    FreeAndNil(ExamIENs); //
                end; //

                FMEExamRequestListBoxClick(Application);
                // Be sure that nothing is still open
                if IsExamOKtoRelease then
                    ReleaseExamToRO; // Release the results to the RO

            end;
        end; { End If Template was for C&P Request } //CodeCR459 JRL 5/21/13

        TIUSignForm.Visible : False;

        if CosigMode True then
            // Find template that matches this document in CAPRI and change
            // status to complete.
            SetMatchingTemplateComplete(TIUNoteIEN)
        else if isCosignatureRequired True then
        begin
            frmMain.RPCBroker1.Results.Clear;

```

```

frmMain.RPCBroker1.Param.Clear;
frmMain.RPCBroker1.RemoteProcedure : 'DVBA CAPRI EXAM LINK TIU';
frmMain.RPCBroker1.Param[0].Value : Piece(PNCSForm.xFMEdit2.IENS, ',', 1);
frmMain.RPCBroker1.Param[0].PType : literal;
frmMain.RPCBroker1.Param[1].Value : TIUNoteIEN;
frmMain.RPCBroker1.Param[1].PType : literal;
frmMain.RPCBrokerCall;
frmMain.RPCBroker1.Call;
if frmMain.RPCBroker1.Results.Count > 0 then
    if frmMain.RPCBroker1.Results[0] <> '1' then
        MessageDlgCAPRI('Error: Unable to link exam to progress note.', mtError, [mbOK], 0);
end;

ModalResult : mrOk; //rra 917335 flag 396.7 for update when returns to TPNCForm.xFormOutputOKClick

{ Send exam to VVA
* Do not send an exam to VVA unless it is an C&P Exam. //CodeCR459 JRL 7/22/13
* Do not send exam if a cosignature is needed (CosignerButton.Caption '')
* CosigMode FALSE, then always send this exam to VVA
* CosigMode TRUE means this exam required a signature and the cosigner
  is reviewing the exam.
* If a cosigner is signing the exam from the popup alert, or if an exam is
  being accessed from the Tools | My Unsigned Exams menu option, this means
  the patient may not have been selected yet and the global variables
  (name/ssn) are not populated. If these variables are blank, then get
  the IEN for the patient stored in ptIENHidden.Caption, retrieve the
  name and ssn, and populate the global variables PatientName and
  PatientSSN so these values exist on the main form when the document is
  sent to VVA. Otherwise, the transmission to VVA will error. Then reset
  the variables so blank so nothing is left behind from the transmit.

Original SendToVVA code: CodeCR423 JRL 11/13/12
Update SendToVVA code handling cosigned exams CodeCR423 JRL 03/08/12
Updated to handle sending to the wrong VVA folder JRL 05/03/13
  because if the same issue as the cosigned exams
  coming from the Alert Popups or the Unsigned Exams
  menu item
Updated to only send C&P Exams to VVA JRL 07/22/13
}

if CnPExam then // CodeCR459 JRL 7/22/13
begin // Only send C&P exams to VVA
    PopulatingNameSSN : FALSE; // default
    SendExamToVVA : FALSE; // default

    SavePatientName : PatientName; // Save the current selected patient
    SavePatientSSN : PatientSSN; // in CAPRI to restore later
    SelectedPatient : TPatient.Create;

    FMGetsVVAPatientInfo.FieldNumbers.Clear; // If a patient is selected from an
    FMGetsVVAPatientInfo.FieldNumbers.Add('01'); // name // alert popup or goes to the unsigned
    FMGetsVVAPatientInfo.FieldNumbers.Add('09'); // ssn // template screen from the tools
    FMGetsVVAPatientInfo.FieldNumbers.Add('02'); // gender
    FMGetsVVAPatientInfo.FieldNumbers.Add('03'); // date of birth format ie, 12/24/1929
    FMGetsVVAPatientInfo.FieldNumbers.Add('991.01'); // icn
    FMGetsVVAPatientInfo.FieldNumbers.Add('313'); // claim number
    FMGetsVVAPatientInfo.FileName : '2'; // menu, the patient exam being signed
    FMGetsVVAPatientInfo.IENS : ptIENHidden.Caption; // may be different from the patient
    FMGetsVVAPatientInfo.GetData; // selected from normal CAPRI use.
    if FMGetsVVAPatientInfo.Results.Count > 0 then //
    begin // Get Selected patient info from
        SelectedPatientName : FMGetsVVAPatientInfo.GetField('01').FMDBExternal; // the IENs stored on the TIU form
        SelectedPatientSSN : FMGetsVVAPatientInfo.GetField('09').FMDBExternal;

//      selectedPatient.FirstName : Piece(SelectedPatientName, ',', 1);
        selectedPatient.LastName : Piece(SelectedPatientName, ',', 1);
        selectedPatient.MiddleName : Piece(Piece(SelectedPatientName, ',', 2), ',', 2);
//      selectedPatient.LastName : Piece(Piece(SelectedPatientName, ',', 2), ',', 1);
        selectedPatient.FirstName : Piece(Piece(SelectedPatientName, ',', 2), ',', 1);

        selectedPatient.Gender : FMGetsVVAPatientInfo.GetField('02').FMDBInternal;
        selectedPatient.Birthdate : FMGetsVVAPatientInfo.GetField('03').FMDBExternal;
        selectedPatient.Birthdate : Piece(selectedPatient.Birthdate, '/', 3) + ' ' +
            Piece(selectedPatient.Birthdate, '/', 1) + ' ' + Piece(selectedPatient.Birthdate, '/', 2);
        selectedPatient.IntegrationControlNumber : FMGetsVVAPatientInfo.GetField('991.01').FMDBExternal;
        selectedPatient.SocialSecurityNumber : SelectedPatientSSN;
        selectedPatient.ClaimNumber : FMGetsVVAPatientInfo.GetField('313').FMDBExternal;
    end
    else
    begin
        SelectedPatientName : '';
        SelectedPatientSSN : '';
    end;

    if (PatientName '') and (PatientSSN '') and //
        (SelectedPatientName '') and (SelectedPatientSSN '') then // We don't know who this is,
    begin // show an error.
        SendExamToVVA : FALSE;
        ShowMessageCapri('This exam must be manually transmitted to Virtual VA!');
    end
    else if (PatientName SelectedPatientName) and (PatientSSN SelectedPatientSSN) then //
    begin // Normal mode name on the
        PopulatingNameSSN : FALSE; // signature screen match name
        SendExamToVVA : TRUE; // signed into CAPRI
    end
end

```

```

else if (PatientName = '') or (PatientSSN = '') then // Probably got here from a
begin // Resolve Alert button since no
    PopulatingNameSSN : TRUE; // CAPRI patient is selected.
    PatientName : SelectedPatientName; // Populate the patient name
    PatientSSN : SelectedPatientSSN; // temporarily before sending
    SendExamToVVA : TRUE; // to VVA
end //
else if (SelectedPatientName <> PatientName) and (SelectedPatientSSN <> PatientSSN) then //
begin // Probably got here from the
    PopulatingNameSSN : TRUE; // Tools Menu | Unsigned Templates.
    PatientName : SelectedPatientName; // This means a CAPRI patient is
    PatientSSN : SelectedPatientSSN; // selected but that is not the
    SendExamToVVA : TRUE; // patient being signed. Populate
// the patient name temporarily for
// sending to VVA
end //
else //
begin //
    SendExamToVVA : FALSE; // We don't know who this is,
    ShowMessageCapri('This exam must be manually transmitted to Virtual VA'); // show an error.
end;

if CosignerButton.Caption <> '' then // Cosignature needed, don't send
    SendExamToVVA : FALSE; // to VVA regardless of code above

if SendExamToVVA then // CodeCR423 JRL 03/08/13
begin
    // Get Report data to automatically send info to the VVA // CodeCR423 JRL 11/13/12
    TIUtoVVAProgressNotes : TStringList.Create; // CodeCR423 JRL 11/13/12
    try // CodeCR423 JRL 11/13/12
        CheckTIUNotes : GetTIUText(TIUNoteIEN, TIUtoVVAProgressNotes); // CodeCR423 JRL 11/13/12
        if CheckTIUNotes then // Notes returned // CodeCR423 JRL 11/14/12
            frmMain.actFileTransmitVirtualVAExecute(Self); // CodeCR423 JRL 11/13/12
        finally // CodeCR423 JRL 11/13/12
            FreeAndNil(TIUtoVVAProgressNotes); // CodeCR423 JRL 11/13/12
        end; // CodeCR423 JRL 11/13/12
    end;
end;

{ if we're sending to VVA then we save to 1) VLER DAS and 2) VistA }
if SendExamToVVA and (isCosignatureRequired = False) then
begin
    if CoSigMode = True then
    begin
        LoadDBQForRendering(cosigTiuDocumentIEN);
        frmMain.NumExamTextForPDFs : pncsForm.Exams.Count;
        SetLength(frmMain.ExamTextForPDFs, frmMain.NumExamTextForPDFs);
        for i : 0 to pncsForm.Exams.Count - 1 do
        begin
            frmMain.ExamTextForPDFs[i] : TMemoryStream.create;
            FinalReport.SaveToStream(frmMain.ExamTextForPDFs[i]);
        end;
    end;
    // Created in Virtual VA code in case it is needed for cosignature text.
    FreeAndNil(FinalReport); // CodeCR565 JRL 7/20/14

    if pncsForm = Nil then
        MessageDlg('DBQ is not loaded; will NOT transmit to VLER DAS', mtError, [mbOk], 0)
    else
    begin
        SendDbqsToVlerAndVista(selectedPatient);
        for i : 1 to pncsForm.Exams.Count do // Setup ReTransmit array // CodeCR562 JRL 7/17/14
            ExamSent[i] : FALSE; // CodeCR562 JRL 7/17/14
        end;
        SendDbqsToVlerAndVista(selectedPatient, FALSE); // Send first time // CodeCR562 JRL 7/17/14
        TransmissionErrors : FALSE; // CodeCR562 JRL 7/17/14
        for i : 1 to pncsForm.Exams.Count do // CodeCR562 JRL 7/17/14
            begin // CodeCR562 JRL 7/17/14
                if ExamSent[i] = FALSE then // CodeCR562 JRL 7/17/14
                    TransmissionErrors : TRUE; // CodeCR562 JRL 7/17/14
                end; // CodeCR562 JRL 7/17/14
            end; // CodeCR562 JRL 7/17/14
        end; // CodeCR562 JRL 7/17/14
        if TransmissionErrors then // CodeCR562 JRL 7/17/14
            begin // CodeCR562 JRL 7/17/14
                if MessageDlgCAPRI('Transmission to VLER failed, do you want to retransmit?', mtConfirmation, [mbYes, mbNo], 0) = mrYes
                then
                    SendDbqsToVlerAndVista(selectedPatient, TRUE); // CodeCR562 JRL 7/17/14
                end; // CodeCR562 JRL 7/17/14
            end;
            // free memory for stored PDFs here // CodeCR562 JRL 7/18/14 moved code to accommodate retransmission //
            CodeCR565 JRL 7/15/14
            for i : 0 to frmMain.NumExamTextForPDFs - 1 do // CodeCR562 JRL 7/18/14 // CodeCR565 JRL 7/15/14
                FreeAndNil(frmMain.ExamTextForPDFs[i]); // CodeCR562 JRL 7/18/14 // CodeCR565 JRL 7/15/14
            end; // CodeCR562 JRL 7/17/14
            if CoSigMode = True then
            begin
                UnloadDBQForRendering;
            end;
        end;

        if PopulatingNameSSN then // if we populated the global variables
        begin // to send the exam to VVA, restore
            PatientName : SavePatientName; // them now to avoid any other issues
            PatientSSN : SavePatientSSN; // in CAPRI
        end;
        FreeAndNil(SelectedPatient);
    end; // if CnPEExam

    ModalResult : mrOk;
end;

```

- 6.2.2.5.61. Dialog - Not applicable for CodeCR565
- 6.2.2.5.62. Help Frame - Not applicable for CodeCR565
- 6.2.2.5.63. HL7 Application Parameter - Not applicable for CodeCR565
- 6.2.2.5.64. HL7 Logical Link - Not applicable for CodeCR565
- 6.2.2.5.65. COTS Interface - Not applicable for CodeCR565
- 6.2.2.6. CODECR562 – DBQ Transmission Failure
 - 6.2.2.6.1. Routines (Entry Points) – Not applicable for CodeCR562
 - 6.2.2.6.2. Templates - Not applicable for CodeCR562
 - 6.2.2.6.3. Bulletins - Not applicable for CodeCR562
 - 6.2.2.6.4. Data Entries Affected by the Design - Not applicable for CodeCR562
 - 6.2.2.6.5. Unique Record(s) - Not applicable for CodeCR562
 - 6.2.2.6.6. File or Global Size Changes - Not applicable for CodeCR562
 - 6.2.2.6.7. Mail Groups - Not applicable for CodeCR562
 - 6.2.2.6.8. Security Keys - Not applicable for CodeCR562
 - 6.2.2.6.9. Options - Not applicable for CodeCR562
 - 6.2.2.6.10. Protocols - Not applicable for CodeCR562
 - 6.2.2.6.11. Remote Procedure Call (RPC) - Not applicable for CodeCR562
 - 6.2.2.6.12. Constants Defined in Interface - Not applicable for CodeCR562
 - 6.2.2.6.13. Variables Defined in Interface - Not applicable for CodeCR562
 - 6.2.2.6.14. Types Defined in Interface - Not applicable for CodeCR562
 - 6.2.2.6.15. GUI

Table 36: GUI

Unit Name	Description
TIUSign.pas	Modified to allow retransmission to VLER of DBQs that failed the first time transmission was attempted.

- 6.2.2.6.16. GUI Classes - Not applicable for CodeCR562
- 6.2.2.6.17. Current Form - Not applicable for CodeCR562
- 6.2.2.6.18. Modified Form - Not applicable for CodeCR562
- 6.2.2.6.19. Components on Form - Not applicable for CodeCR562
- 6.2.2.6.20. Events

Table 40: Events

Name	Type	Description
ButtonOK2Click	Event	Modified routine to allow retransmission of DBQs that failed to send the first time.

- 6.2.2.6.21. Methods - Not applicable for CodeCR562
- 6.2.2.6.22. Special References - Not applicable for CodeCR562
- 6.2.2.6.23. Class Events - Not applicable for CodeCR562
- 6.2.2.6.24. Class Methods - Not applicable for CodeCR562
- 6.2.2.6.25. Class Properties - Not applicable for CodeCR562
- 6.2.2.6.26. Uses Clause - Not applicable for CodeCR562
- 6.2.2.6.27. Forms - Not applicable for CodeCR562
- 6.2.2.6.28. Functions

Table 48: Functions

Function Name	SendDbqsToVlerAndVista
Short Description	Modified to allow retransmission to VLER of DBQs that failed the first time transmission was attempted.
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	N/A	N/A

Function Name	SendDbqsToVlerAndVista
Data Dictionary (DD) References	N/A

Function Name	SendDbqsToVlerAndVista
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input checked="" type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name:ReTransmit Definition: Boolean
Output Attribute Name and Definition	Name:N/A Definition:N/A

Current Logic

```

procedure TTIUSignForm.SendDbqsToVlerAndVista(selectedPatient: TPatient; ReTransmit : Boolean);
const
    WEB_SERVICE_BUNDLE    'WebServiceBundle';
    EXTENSION_ZIP         '.zip';
    EXTENSION_UUE         '.uue';
var
    i: Integer;

    singleExam: TCPWMEExam;
    claimAttachment: TVlerDasAttachment;
    xmlEngine: TXmlEngine;

    xmlDocument: IXMLDocument;
    xmlDocumentNoAttachments: IXMLDocument;

    vlerDasClient: TvlerDasClient;
    vlerDasThread: TVlerDasClientThread;
    vlerDasClientResult: TStringList;

    savedExamFiles: TStringList;
    tempFile: String;
    CookieMgr : TIdCookieManager;
    vlerSplitExam : TSplitExam; // CodeCR565 JRL 7/10/14
    CnPTitle : String; // CodeCR565 JRL 7/13/14
    CurrentDateTime, DateOfNote : String; // CodeCR565 JRL 7/14/14
    aSignature, buffer : String; // CodeCR565 JRL 7/15/14
    dt,tm : String; // CodeCR565 JRL 7/15/14

begin

    // Get the TIU Signature just created
    frmMain.RPCBroker1.Results.Clear; // CodeCR565 JRL 7/15/14
    frmMain.RPCBroker1.RemoteProcedure : 'TIU GET ADDITIONAL SIGNERS'; // CodeCR565 JRL 7/15/14
    frmMain.RPCBroker1.Param[0].Value : TIUNoteIEN; // CodeCR565 JRL 7/15/14
    frmMain.RPCBroker1.Param[0].PType : literal; // CodeCR565 JRL 7/15/14
    frmMain.RPCBrokerCall; // CodeCR565 JRL 7/15/14
    Try // CodeCR565 JRL 7/15/14
        frmMain.RPCBroker1.Call; // CodeCR565 JRL 7/15/14
    Except // CodeCR565 JRL 7/15/14
        On EBrokerError Do // CodeCR565 JRL 7/15/14
            ShowMessageCAPRI('RPC TIU GET ADDITIONAL SIGNERS could not be accessed!'); // CodeCR565 JRL 7/15/14
    End; // CodeCR565 JRL 7/15/14
    If frmMain.RPCBroker1.Results.Count > 0 Then // CodeCR565 JRL 7/15/14
        begin // CodeCR565 JRL 7/15/14
            // rearrange name from LAST,FIRST M to FIRST M LAST // CodeCR565 JRL 7/15/14
            buffer : Piece(frmMain.RPCBroker1.Results[0], '^', 2); // CodeCR565 JRL 7/15/14
            i: Pos(',',buffer); // 5 // CodeCR565 JRL 7/15/14
            aSignature : copy(buffer,i+1,length(buffer) i) + ' ' + copy(buffer,1,i 1); // CodeCR565 JRL 7/15/14
        end // CodeCR565 JRL 7/15/14
    Else // CodeCR565 JRL 7/15/14
        aSignature : ''; // CodeCR565 JRL 7/15/14
    vlerSplitExam : TSplitExam.Create; // CodeCR565 JRL 7/10/14
    CnPTitle : PNTitles.Items[PNTitles.ItemIndex]; // CodeCR565 JRL 7/13/14
    if CallRPC(frmMain.RPCBroker1, 'ORWU DT', ['NOW'], nil, False, True) then // CodeCR565 JRL 7/13/14

```

```

begin
    buffer : frmMain.RPCBroker1.Results[0]; // CodeCR565 JRL 7/15/14
    dt : Piece(buffer, '.', 1); // CodeCR565 JRL 7/13/14
    tm : Piece(buffer, '.', 2); // CodeCR565 JRL 7/15/14
    CurrentDateTime : Copy(dt, 4, 2) + '/' + Copy(dt, 6, 2) + '/20' + Copy(dt, 2, 2) + // CodeCR565 JRL 7/15/14
    ' ' + Copy(tm, 1, 2) + ':' + Copy(tm, 3, 2) + ':' + Copy(tm, 5, 2); // CodeCR565 JRL 7/15/14
end; // CodeCR565 JRL 7/15/14

if CheckBoxOpinion.Checked then // CodeCR565 JRL 7/15/14
    DateOfNote : CurrentDateTime // CodeCR565 JRL 7/15/14
else // CodeCR565 JRL 7/16/14
begin // CodeCR565 JRL 7/15/14
    if ListBox1.ItemIndex <> 1 then // CodeCR565 JRL 7/15/14
        DateOfNote : ListBox1.Items[Listbox1.ItemIndex] // Appointments // CodeCR565 JRL 7/15/14
    else if ListBox4.ItemIndex <> 1 then // CodeCR565 JRL 7/16/14
        DateOfNote : ListBox4.Items[Listbox4.ItemIndex] // Admissions // CodeCR565 JRL 7/15/14
    else // CodeCR565 JRL 7/16/14
        DateOfNote : CurrentDateTime; // Default // CodeCR565 JRL 7/15/14
    end; // CodeCR565 JRL 7/16/14
vlerSplitExam.ProcessTextToPDF(aSignature, CurrentDateTime, FMExProvider.Text, // CodeCR565 JRL 7/15/14
    CosignerButton.Caption, CnPTitle, DateOfNote); // CodeCR565 JRL 7/15/14
FreeAndNil(vlerSplitExam); // CodeCR565 JRL 7/10/14

frmMain.VlerDasToken : GetRemoteAuthenticationToken(frmMain.RPCBroker1); // CodeCR540 JRL 12/11/13
savedExamFiles : TStringList.Create;

// create partial claim attachment object; remaining completed once xml engine is available
CookieMgr : TIdCookieManager.Create(); //CodeCR540 JRL 12/12/13
for i : 0 to pncsForm.Exams.Count - 1 do
begin
    if (frmTIUCosign <> Nil) and (frmTIUCosign.Showing) then
        frmTIUCosign.Repaint
    else if (pncsForm <> Nil) and (pncsForm.Showing) then
        pncsForm.Repaint;

    singleExam : pncsForm.Exams[i];

    if singleExam.IsXmlBasedExam False then // skip exams without an XSD
        continue;

    // Get single PDF from memory
    PdfMemoryStream.Position : 0; // CodeCR565 JRL 7/13/14
    PdfMemoryStream.LoadFromStream(frmMain.ExamTextForPDFs[i]); //Get Exam Text// CodeCR565 JRL 7/13/14

    if (PdfMemoryStream <> Nil) and (PdfMemoryStream.Size > 0) then // moved this code into the loop instead
begin // of happening before loop CodeCR565 JRL 7/10/14
    claimAttachment : TVlerDasAttachment.Create;
    claimAttachment.BinaryBase64Object : claimAttachment.EncodeToBase64(PdfMemoryStream);
    claimAttachment.BinaryDescriptionText : 'text is replaced below';
    claimAttachment.BinaryFormatStandardName : 'application/pdf';
    claimAttachment.BinaryLocationURI : 'text is replaced below';
    claimAttachment.BinarySizeValue : IntToStr(PdfMemoryStream.Size);
    claimAttachment.BinaryCategoryText : 'text is replaced below';
end;

xmlEngine : TXmlEngine.Create(singleExam);

if claimAttachment <> Nil then
begin
    claimAttachment.BinaryDescriptionText : xmlEngine.GetDocumentTitleText;
    claimAttachment.BinaryCategoryText : xmlEngine.GetDocumentTitleText;
    claimAttachment.BinaryLocationURI : xmlEngine.GetUniqueFilename(xmlEngine.GetDocumentTitleText, 'pdf');
    xmlEngine.AddClaimAttachment(claimAttachment);
end;

xmlEngine.SetPatient(selectedPatient);
xmlEngine.PopulateApproverInfo('','','',''); // CodeCR563 JRL 6/13/14
xmlEngine.PopulateExamInfo('VHA_CAPRI','','Inhouse','','xmlEngine.GetW3cDateTime,','Completed'); // CodeCR563 JRL 6/13/14

xmlDocument : xmlEngine.RenderToXml(pncsForm.xpanelBaseControl);
xmlDocumentNoAttachments : xmlEngine.RenderToXml(pncsForm.xpanelBaseControl, False);

if PARAM_SAVE_XML True then
begin
    xmlDocument.SaveToFile(GetCurrentDir + '\exam request ' + IntToStr(i) + '.xml');
    xmlDocumentNoAttachments.SaveToFile(GetCurrentDir + '\exam request noa ' + IntToStr(i) + '.xml');
end;

tempFile : xmlEngine.GetTempDirectory + 'exam request ' + IntToStr(i) + '.xml';
if FileExists(tempFile) then
    DeleteFile(tempFile);
xmlDocumentNoAttachments.SaveToFile(tempFile);
savedExamFiles.Add(tempFile);

{ send payload to vler das }
try
    vlerDasClient : TfVlerDasClient.Create(Self);
    vlerDasClient.SetDBQName(xmlEngine.GetDocumentTitleText);
    vlerDasClient.SetCookieManager(CookieMgr);

    vlerDasThread : TVlerDasClientThread.Create(vlerDasClient);

    vlerDasClientResult : TStringList.Create;
    vlerDasThread.VlerDasClientResult : vlerDasClientResult;

```

```

try
    vlerDasThread.XmlDocument : xmlDocument;
    vlerDasThread.DocumentIdentificationID : xmlEngine.GetDocumentIdentificationID;
    vlerDasThread.Resume;
    vlerDasThread.WaitFor;
    ExamSent[i+1] : CheckErrorsForRetransmitOption(vlerDasClientResult);
    if vlerDasThread.AnException <> Nil then
        raise vlerDasThread.AnException;
    except
        on E: Exception do
            begin
                if vlerDasClientResult.Text <> '' then
                    vlerDasClientResult.Text : E.Message;
                    MessageDlg('Failed to transmit the DBQ XML for ' +
                        xmlEngine.GetDocumentTitleText + #10#13#10#13 + 'Reason: ' +
                        E.Message, mtError, [mbOK], 0);
                end;
            end;
        end;

    if PARAM_SAVE_XML True then
        vlerDasClientResult.SaveToFile(GetCurrentDir + '\exam response ' + IntToStr(i) + '.xml');

    tempFile : xmlEngine.GetTempDirectory + 'exam response ' + IntToStr(i) + '.xml';
    if FileExists(tempFile) then
        DeleteFile(tempFile);
    vlerDasClientResult.SaveToFile(tempFile);

    savedExamFiles.Add(tempFile);

    FreeAndNil(vlerDasThread);
finally
    Screen.Cursor : crDefault;
end;
FreeAndNil(vlerDasClient);
FreeAndNil(xmlEngine);
end;
FreeAndNil(CookieMgr); //CodeCR540 JRL 12/12/13

if savedExamFiles.Count > 0 then
begin
    { bundle to include request and response status for each webservice invocation }
    MiniZip1.Zipfile : xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP;
    if FileExists(MiniZip1.Zipfile) then
        DeleteFile(MiniZip1.Zipfile);

    for i : 0 to savedExamFiles.Count - 1 do
    begin
        MiniZip1.AddToZipFile(savedExamFiles[i], ExtractFileName(savedExamFiles[i]));
    end;

    { uu encode the zip file }
    if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE) then
        DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);

    UUEncode(MiniZip1.Zipfile);

    { send the payload bundle to vista }
    SendDbqsToVista(parentExamIen, xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);

    { clean up our mess }
    for i : 0 to savedExamFiles.Count - 1 do
    begin
        if FileExists(savedExamFiles[i]) then
            DeleteFile(savedExamFiles[i]);
    end;

    if PARAM_SAVE_XML True then
    begin
        if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP) then
            CopyFile(PAnsiChar(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP),
                PAnsiChar(GetCurrentDir + '\ ' + WEB_SERVICE_BUNDLE + EXTENSION_ZIP), False);
        end;

        if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP) then
            DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP);

        if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE) then
            DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);

    end; // if savedExamFiles.Count > 0 then

    if PdfMemoryStream <> Nil then
        FreeAndNil(PdfMemoryStream);

    if claimAttachment <> Nil then
        FreeAndNil(claimAttachment);

    if savedExamFiles <> Nil then
        FreeAndNil(savedExamFiles);

end;

```

Modified Logic (Changes are in bold)

```

procedure TTIUSignForm.SendDbqsToVlerAndVista(selectedPatient: TPatient; ReTransmit : Boolean);
const
    WEB SERVICE BUNDLE = 'WebServiceBundle';
    EXTENSION ZIP = '.zip';
    EXTENSION UUE = '.uue';
var
    i: Integer;

    singleExam: TCPWMEExam;
    claimAttachment: TVlerDasAttachment;
    xmlEngine: TXmlEngine;

    xmlDocument: IXMLDocument;
    xmlDocumentNoAttachments: IXMLDocument;

    vlerDasClient: TfVlerDasClient;
    vlerDasThread: TVlerDasClientThread;
    vlerDasClientResult: TStringList;

    savedExamFiles: TStringList;
    tempFile: String;
    CookieMgr : TIdCookieManager;
    vlerSplitExam : TSplitExam; // CodeCR565 JRL 7/10/14
    CnPTitle : String; // CodeCR565 JRL 7/13/14
    CurrentDateTime, DateOfNote : String; // CodeCR565 JRL 7/14/14
    aSignature, buffer : String; // CodeCR565 JRL 7/15/14
    dt,tm : String; // CodeCR565 JRL 7/15/14
    ErrorMsg : string; // CodeCR562 JRL 06/25/14
    TransmitAgain : Boolean; // CodeCR562 JRL 7/16/14

begin

    If (ReTransmit = FALSE) and // setup the files the first time only // CodeCR562 JRL 7/17/14
    (CoSigMode = FALSE) then // skip setup, file is already complete from vva // CodeCR565 JRL 7/20/14
    begin // CodeCR562 JRL 7/17/14
        // Get the TIU Signature just created
        frmMain.RPCBroker1.Results.Clear; // CodeCR565 JRL 7/15/14
        frmMain.RPCBroker1.RemoteProcedure := 'TIU GET ADDITIONAL SIGNERS'; // CodeCR565 JRL 7/15/14
        frmMain.RPCBroker1.Param[0].Value := TIUNoteIEN; // CodeCR565 JRL 7/15/14
        frmMain.RPCBroker1.Param[0].PType := literal; // CodeCR565 JRL 7/15/14
        frmMain.RPCBrokerCall; // CodeCR565 JRL 7/15/14
        Try // CodeCR565 JRL 7/15/14
            frmMain.RPCBroker1.Call; // CodeCR565 JRL 7/15/14
        Except // CodeCR565 JRL 7/15/14
            On EBrokerError Do // CodeCR565 JRL 7/15/14
                ShowMessageCAPRI('RPC TIU GET ADDITIONAL SIGNERS could not be accessed!'); // CodeCR565 JRL 7/15/14
        End; // CodeCR565 JRL 7/15/14
        If frmMain.RPCBroker1.Results.Count > 0 Then // CodeCR565 JRL 7/15/14
            begin // CodeCR565 JRL 7/15/14
                // rearrange name from LAST,FIRST M to FIRST M LAST // CodeCR565 JRL 7/15/14
                buffer := Piece(frmMain.RPCBroker1.Results[0], '^', 2); // CodeCR565 JRL 7/15/14
                i:=Pos(',',buffer); // 5 // CodeCR565 JRL 7/15/14
                aSignature := copy(buffer,i+1,length(buffer) i) + ' ' + copy(buffer,1,i 1); // CodeCR565 JRL 7/15/14
            end // CodeCR565 JRL 7/15/14
            Else // CodeCR565 JRL 7/15/14
                aSignature := ''; // CodeCR565 JRL 7/15/14
            vlerSplitExam := TSplitExam.Create; // CodeCR565 JRL 7/10/14
            CnPTitle := PNTitles.Items[PNTitles.ItemIndex]; // CodeCR565 JRL 7/13/14
            if CallRPC(frmMain.RPCBroker1, 'ORWU DT', ['NOW'], nil, False, True) then // CodeCR565 JRL 7/13/14
                begin // CodeCR565 JRL 7/15/14
                    buffer := frmMain.RPCBroker1.Results[0]; // CodeCR565 JRL 7/13/14
                    dt := Piece(buffer, '.',1); // CodeCR565 JRL 7/15/14
                    tm := Piece(buffer, '.',2); // CodeCR565 JRL 7/15/14
                    CurrentDateTime := Copy(dt,4,2) + '/' + Copy(dt,6,2) + '/20' + Copy(dt,2,2) + // CodeCR565 JRL 7/15/14
                        ' ' + Copy(tm,1,2) + ':' + Copy(tm,3,2) + ':' + Copy(tm,5,2); // CodeCR565 JRL 7/15/14
                end; // CodeCR565 JRL 7/15/14

            if CheckBoxOpinion.Checked then // CodeCR565 JRL 7/15/14
                DateOfNote := CurrentDateTime // CodeCR565 JRL 7/15/14
            else // CodeCR565 JRL 7/16/14
                begin // CodeCR565 JRL 7/15/14
                    if ListBox1.ItemIndex <> 1 then // CodeCR565 JRL 7/15/14
                        DateOfNote := ListBox1.Items[ListBox1.ItemIndex] // Appointments // CodeCR565 JRL 7/15/14
                    else if ListBox4.ItemIndex <> 1 then // CodeCR565 JRL 7/16/14
                        DateOfNote := ListBox4.Items[ListBox4.ItemIndex] // Admissions // CodeCR565 JRL 7/15/14
                    else // CodeCR565 JRL 7/16/14
                        DateOfNote := CurrentDateTime; // Default // CodeCR565 JRL 7/15/14
                    end; // CodeCR565 JRL 7/16/14
                    vlerSplitExam.ProcessTextToPDF(aSignature,CurrentDateTime, FMExProvider.Text, // CodeCR565 JRL 7/15/14

```

```

        CosignerButton.Caption,CnPTitle,DateOfNote); // CodeCR565 JRL 7/15/14
    FreeAndNil(vlerSplitExam); // CodeCR565 JRL 7/10/14
end; // if ReTransmit = FALSE // CodeCR562 JRL 7/17/14

frmMain.VlerDasToken := GetRemoteAuthenticationToken(frmMain.RPCBroker1); // CodeCR540 JRL 12/11/13
savedExamFiles := TStringList.Create;

// create partial claim attachment object; remaining completed once xml engine is available
PdfMemoryStream := TMemoryStream.Create; //CodeCR562 JRL 7/18/14
CookieMgr := TIdCookieManager.Create(); //CodeCR540 JRL 12/12/13
for i := 0 to pncsForm.Exams.Count - 1 do
begin
    if (frmTIUCosign <> Nil) and (frmTIUCosign.Showing) then
        frmTIUCosign.Repaint
    else if (pncsForm <> Nil) and (pncsForm.Showing) then
        pncsForm.Repaint;

    singleExam := pncsForm.Exams[i];

    if singleExam.IsXmlBasedExam = False then // skip exams without an XSD
    begin
        ExamSent[i+1] := TRUE; // make sure legacy exams show as being sent so they don't throw off error count
        //CodeCR562 JRL 7/18/14
        continue;
    end;
    if ReTransmit and ExamSent[i+1] = TRUE then // skip sending an exam that was already sent correctly //
        //CodeCR562 JRL 7/17/14
        continue; // CodeCR562 JRL 7/17/14

    // Get single PDF from memory
    PdfMemoryStream.Position := 0; // CodeCR565 JRL 7/13/14
    PdfMemoryStream.LoadFromStream(frmMain.ExamTextForPDFs[i]); //Get Exam Text// CodeCR565 JRL 7/13/14

    if (PdfMemoryStream <> Nil) and (PdfMemoryStream.Size > 0) then // moved this code into the loop instead
    begin // of happening before loop CodeCR565 JRL
        7/10/14
        claimAttachment := TVlerDasAttachment.Create;
        claimAttachment.BinaryBase64Object := claimAttachment.EncodeToBase64(PdfMemoryStream);
        claimAttachment.BinaryDescriptionText := 'text is replaced below';
        claimAttachment.BinaryFormatStandardName := 'application/pdf';
        claimAttachment.BinaryLocationURI := 'text is replaced below';
        claimAttachment.BinarySizeValue := IntToStr(PdfMemoryStream.Size);
        claimAttachment.BinaryCategoryText := 'text is replaced below';
    end;

    xmlEngine := TXmlEngine.Create(singleExam);

    if claimAttachment <> Nil then
    begin
        claimAttachment.BinaryDescriptionText := xmlEngine.GetDocumentTitleText;
        claimAttachment.BinaryCategoryText := xmlEngine.GetDocumentTitleText;
        claimAttachment.BinaryLocationURI := xmlEngine.GetUniqueFilename(xmlEngine.GetDocumentTitleText, 'pdf');
        xmlEngine.AddClaimAttachment(claimAttachment);
    end;

    xmlEngine.SetPatient(selectedPatient);
    xmlEngine.PopulateApproverInfo('','','',''); // CodeCR563 JRL 6/13/14
    xmlEngine.PopulateExamInfo('VHA CAPRI','','Inhouse','',xmlEngine.GetW3cDateTime(),'','Completed'); //
    CodeCR563 JRL 6/13/14

    xmlDocument := xmlEngine.RenderToXml(pncsForm.xpanelBaseControl);
    xmlDocumentNoAttachments := xmlEngine.RenderToXml(pncsForm.xpanelBaseControl, False);

    if PARAM SAVE XML = True then
    begin
        xmlDocument.SaveToFile(GetCurrentDir + '\exam request ' + IntToStr(i) + '.xml');
        xmlDocumentNoAttachments.SaveToFile(GetCurrentDir + '\exam request noa ' + IntToStr(i) + '.xml');
    end;

    tempFile := xmlEngine.GetTempDirectory + 'exam request ' + IntToStr(i) + '.xml';
    if FileExists(tempFile) then
        DeleteFile(tempFile);
    xmlDocumentNoAttachments.SaveToFile(tempFile);
    savedExamFiles.Add(tempFile);

    { send payload to vler das }
    try
        vlerDasClient := TvlerDasClient.Create(Self);
        vlerDasClient.SetDBQName(xmlEngine.GetDocumentTitleText);
        vlerDasClient.SetCookieManager(CookieMgr);

```

```

vlerDasThread := TVlerDasClientThread.Create(vlerDasClient);

vlerDasClientResult := TStringList.Create;
vlerDasThread.VlerDasClientResult := vlerDasClientResult;

try
    vlerDasThread.XmlDocument := xmlDocument;
    vlerDasThread.DocumentIdentificationID := xmlEngine.GetDocumentIdentificationID;
    vlerDasThread.Resume;
    vlerDasThread.WaitFor;
    ExamSent[i+1] := CheckErrorsForRetransmitOption(vlerDasClientResult);
    if vlerDasThread.AnException <> Nil then
        raise vlerDasThread.AnException;
    except
        on E: Exception do
            begin
                ExamSent[i+1] := FALSE; // exam didn't send
            end;
        end;
    end;

    if PARAM SAVE XML = True then
        vlerDasClientResult.SaveToFile(GetCurrentDir + '\exam response ' + IntToStr(i) + '.xml');

    tempFile := xmlEngine.GetTempDirectory + 'exam response ' + IntToStr(i) + '.xml';
    if FileExists(tempFile) then
        DeleteFile(tempFile);
    vlerDasClientResult.SaveToFile(tempFile);

    savedExamFiles.Add(tempFile);

    FreeAndNil(vlerDasThread);
finally
    Screen.Cursor := crDefault;
end;
FreeAndNil(vlerDasClient);
FreeAndNil(xmlEngine);
end;
FreeAndNil(CookieMgr); //CodeCR540 JRL 12/12/13

if savedExamFiles.Count > 0 then
begin
    { bundle to include request and response status for each webservice invocation }
    MiniZip1.Zipfile := xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP;
    if FileExists(MiniZip1.Zipfile) then
        DeleteFile(MiniZip1.Zipfile);

    for i := 0 to savedExamFiles.Count - 1 do
    begin
        MiniZip1.AddToZipFile(savedExamFiles[i], ExtractFileName(savedExamFiles[i]));
    end;

    { uu encode the zip file }
    if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE) then
        DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);

    UUEncode(MiniZip1.Zipfile);

    { send the payload bundle to vista }
    SendDbqsToVista(parentExamIen, xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);

    { clean up our mess }
    for i := 0 to savedExamFiles.Count - 1 do
    begin
        if FileExists(savedExamFiles[i]) then
            DeleteFile(savedExamFiles[i]);
    end;

    if PARAM SAVE XML = True then
    begin
        if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP) then
            CopyFile(PAnsiChar(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP),
                PAnsiChar(GetCurrentDir + '\ ' + WEB_SERVICE_BUNDLE + EXTENSION_ZIP), False);
    end;

    if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP) then
        DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP);

    if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE) then
        DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);
end;

```

```

end; // if savedExamFiles.Count > 0 then

if PdfMemoryStream <> Nil then
    FreeAndNil(PdfMemoryStream);

if claimAttachment <> Nil then
    FreeAndNil(claimAttachment);

if savedExamFiles <> Nil then
    FreeAndNil(savedExamFiles);

end;

```

Table 48: Functions

Function Name	ButtonOK2Click
Short Description	Modified to allow retransmission to VLER of DBQs that failed the first time transmission was attempted.
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	N/A

Function Name	ButtonOK2Click
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition: N/A
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic

```

//
// OK Button    Sign the exam

```

```

//
// YES/NO    this is a C&P Exam notes
//
// If the user chooses NO to "is this a C&P exam" and there is a cosignature,
// handle the same way as "NO" without a cosignature, i.e. mark the exam as
// complete. In addition, validate the cosigner data, and then delete it. This
// avoids alerts being sent to the cosigner (desired) for non C&P exams.
//
// If the user chooses NO to "is this a C&P exam", do not send that note to VVA
// or VLER/DAS web service. Only C&P exams are sent to any webservises now.
//
// Check to see if the question "Is this a C&P exam" is begin displayed.
// If so, handle normally. If it is not being displayed, then the user is
// coming from an alert/cosignature screen. Treat the exam as a C&P exam.
//
procedure TTIUSignForm.ButtonOK2Click(Sender: TObject);
var
    Appointment: String;
    TitleIEN: String;
    ErrMsg: String;
    CheckTIUNotes : Boolean;
    SendExamToVVA : Boolean;           // CodeCR423 JRL 3/8/13
    PopulatingNameSSN : Boolean;       // CodeCR423 JRL 3/8/13
    I : Integer;                      // CodeCR471 JRL 4/30/13
    ExamIENs : TStrings;              // CodeCR471 JRL 4/30/13
    RequestIEN : String;              // CodeCR471 JRL 5/08/13
    buffer : string;                  // CodeCR471 JRL 5/08/13
    SelectedPatientName, SavePatientName : String; // JRL 5/3/13
    SelectedPatientSSN, SavePatientSSN : String;   // JRL 5/3/13
    CosignYes CnPExamNO : Boolean;    // CodeCR459 JRL 7/18/13
    CnPExam : Boolean;                // CodeCR459 JRL 7/22/13
    selectedPatient: TPatient;

    {cosignature variables needed for vler das #186}
    isCosignatureRequired :Boolean;
    cosigTiuDocumentIEN: String;
    cosigCapriTemplateIEN: String;

begin
    { is a cosignature required? }
    isCosignatureRequired := Trim(FMEdit16.Text) <> '';

    if Self.Owner is TfrmTIUCosign then
        cosigTiuDocumentIEN := (Self.Owner as TfrmTIUCosign).TiuDocumentIEN;

    { identify which exam will be the parent exam ien, addresses an edge case bug }
    for i := 0 To lstExamsRequested.Items.Count - 1 do
        begin
            if lstExamsRequested.Selected[i] = true then
                begin
                    parentExamIen := Piece(lstExamsRequested.Items[i], '^', 2);
                    ParentExamTitle := TrimRight(Piece(lstExamsRequested.Items[i], '^', 1)); // JRL 09/27/13 C&P Exam Title
                    break;
                end;
            end;
        end;

    // Is the question "Is this a C&P exam" being displayed?
    if cmbxCPRqst.Visible then                                     //CodeCR459 JRL 7/22/13
        CnPExam := SameText(UpperCase(cmbxCPRqst.Text), 'YES')      //CodeCR459 JRL 5/21/13
    else                                                             //CodeCR459 JRL 7/22/13
        CnPExam := TRUE;                                           //CodeCR459 JRL 7/22/13

    // If the user chooses NO to "is this a C&P exam" and there is a cosignature,
    // handle the same way as "NO" without a cosignature, i.e. mark the exam as
    // complete.
    if (CnPExam = FALSE) and                                       //CodeCR459 JRL 7/22/13
        (cosignerButton.Caption <> '') then                         //CodeCR459 JRL 7/18/13
        CosignYes CnPExamNO := TRUE                                 //CodeCR459 JRL 7/18/13
    else                                                             //CodeCR459 JRL 7/18/13
        CosignYes CnPExamNO := FALSE;                             //CodeCR459 JRL 7/18/13

    ComboBoxExamLocation.ItemIndex := 0;

    // Set to release report in AMIE by default.
    ComboBoxReleaseReport.ItemIndex := 0;

    CosigMode := False;
    if CoSigTIUNoteIEN <> '' then
        begin
            TIUNoteIEN := CoSigTIUNoteIEN;

```



```

    CosigMode := True;
end;

//Validate user input fields  Short circuit on failure
if NOT IsUserInputValid(CosigMode) then
    Exit;

// wait until validation is done. Then check if the condition of a Non C&P
// Exam is being signed that requires a cosignature. If so, delete the
// cosignature information and send the TIU note only. Removing the cosignature
// information keeps the cosigner from getting an "alert" when they login to
// CAPRI. The cosignature process is handled outside of CAPRI for non C&P
// exams so we don't care who the cosigner was supposed to be. Bottom line is
// sacrificing storing this data to avoid the alert.
if CosignYes CnPExamNO = TRUE then
begin
    cosignerButton.Caption := '';
    FMEdit16.Text := ''; // Clear out saved IEN
end;

//Are any exams pending  Short circuit if NO
if CnPExam then //CodeCR459 JRL 5/21/13
begin
    if FMEExamRequestListbox.Items.Count = 0 then
    begin
        ShowMessageCAPRI('There are no AMIE exam requests needing results. This document must be linked to an
open request before it can be uploaded.');
```

ModalResult := mrCancel;

Exit;

end;

end; { End If Template was for C&P Request }

```

if CosigMode = False then
begin {Start NOT CosigMode}
    {===== Swap date and clinic location =====}
    VisitIDIEN := ''; // clear any previous visit IENS
    if Listbox4.ItemIndex > 1 then
        VisitIDIEN := Piece(Listbox5.Items[Listbox4.ItemIndex], ';', 2) + ';'
        + Piece(Listbox5.Items[Listbox4.ItemIndex], ';', 1) + ';H'
    else if Listbox1.ItemIndex > 1 then
    begin
        Appointment := FindMatchingApptData;
        VisitIDIEN := Piece(Appointment, '^', 2) + ';'
        + Piece(Appointment, '^', 1) + ';A';
    end;

    {===== Create and Populate TIU Record =====}
    if TIUNoteIEN = '' then
    begin
        TitleIEN := GetTIUTitleIEN;
        TIUNoteIEN := CreateTIURecord(VisitIDIEN, TitleIEN);

        if Piece(TIUNoteIEN, '^', 1) = '0' then
        begin
            ShowMessageCAPRI('Could not create TIU Record. Cannot Continue. ' + Piece(TIUNoteIEN, '^', 2));
            ModalResult := mrCancel;
            Exit;
        end;

        // Populate TIU record text
        ErrMsg := '';
        if PNCSForm.xSampleReportOutput.Lines.Count > 0 then
            SetText(ErrMsg, PNCSForm.xSampleReportOutput.Lines, StrToInt(TIUNoteIEN), 0);

        if ErrMsg <> '' then
            ShowMessageCAPRI(ErrMsg);
    end;

    PNCSForm.xPanelSave.Visible := False;
    PNCSForm.xGauge1.Progress := 0;
    TIUSignForm.Visible := True;

    {===== Validate TIU Text transfer =====}
    if NOT IsTIUTextValid(TIUNoteIEN) then
    begin
        ModalResult := mrCancel;
        Exit;
    end;

    {==== Sign TIU Record and if failed, then try to delete TIU ====}
    if SignTIURecord(TIUNoteIEN, Edit3.Text) then

```

```

begin
    //Update status and TIU Document # in CAPRI and AMIE files
    PNCSForm.XFMEEdit10.Text := TIUNoteIEN; // Set TIU Document number in CAPRI template file
    // Set Review Status
    if cosignerButton.Caption <> '' then
        pncsForm.xFMEEditReviewStatus.Text := 'U'
    else
        pncsForm.xFMEEditReviewStatus.Text := 'C';
        if CosignYes CnPExamNO then // CodeCR459 JRL 7/22/13 If cosigning a non c&p exam,
then set
            pncsForm.xFMEEditReviewStatus.Text := 'C'; // CodeCR459 JRL 7/22/13 the status to complete.
            pncsForm.xFMEEdit8.Text := 'NOW'; {Set date/time of sig to lock note in IPR file}
        end
    else
        begin
            DeleteTIURecord(TIUNoteIEN, '');
            ModalResult := mrCancel;
            Exit;
        end;
    end {End NOT CosigMode}
else
    begin {Start CosigMode}
        if SignTIURecord(TIUNoteIEN, Edit3.Text) then
            begin
                if frmTIUCosign.addendumText.Lines.Count > 0 then
                    frmTIUCosign.SaveAddendum;
                end
            else
                Exit;
            end;
        end {End CosigMode}

        // Get Epsiode Date/Time
        anuApptPointer := ''; //CodeCR499 JRL 6/27/13
        if not CheckBoxOpinion.Checked then //CodeCR499 JRL 6/27/13
            anuApptPointer := GetApptPointer(TIUNoteIEN);

        if CnPExam then //CodeCR459 JRL 7/22/13
            begin //Added if statement and
encapsulated //existing code

                // Set AMIE with TIU text
                if labelCosigner.enabled = false then
                    SetAMIEResults(TIUNoteIEN, VisitIDIEN, anuApptPointer);

                //Store linked AMIE exams in CAPRI Template file
                SetExamInTemplate(coSigMode);

                if labelCosigner.enabled = false then
                    begin
                        // Send out email for completed exam(s) // Added email send routine
                        try // from try/finally inclusive
                            ExamIENS := TStringList.Create; // CodeCR471 JRL 5/1/13
                            ExamIENS.Clear; // Relocated code JRL 6/25/13
                            for I := 0 To lstExamsRequested.Items.Count - 1 do //
                                begin //
                                    if lstExamsRequested.Selected[I] = true then // Get selected exam IEN
                                        begin // and add to exam list
                                            buffer := Piece(lstExamsRequested.Items[I], '^', 2); //
                                            ExamIENS.Add(buffer); //
                                        end; //
                                    end; //
                                end; //
                                RequestIEN := FMEExamRequestListbox.GetSelectedRecord.IEN; //
                                if NOT SendNotificationTwo(ExamIENS, //
                                    AuthorIEN, //
                                    RequestIEN) then //
                                    ShowMessageCAPRI('There was a problem sending a completion email notification. Please notify
IRM!');
                                finally //
                                    FreeAndNil(ExamIENS); //
                                end; //

                                FMEExamRequestListboxClick(Application);
                                // Be sure that nothing is still open
                                if IsExamOKtoRelease then
                                    ReleaseExamToRO; // Release the results to the RO

                                end;
                            end; { End If Template was for C&P Request } //CodeCR459 JRL 5/21/13

                            TIUSignForm.Visible := False;

```

```

if CosigMode = True then
    // Find template that matches this document in CAPRI and change
    // status to complete.
    SetMatchingTemplateComplete(TIUNoteIEN)
else if isCosignatureRequired = True then
begin
    frmMain.RPCBroker1.Results.Clear;
    frmMain.RPCBroker1.Param.Clear;
    frmMain.RPCBroker1.RemoteProcedure := 'DVBA CAPRI EXAM LINK TIU';
    frmMain.RPCBroker1.Param[0].Value := Piece(PNCSForm.xFMEdit2.IENS, ',', 1);
    frmMain.RPCBroker1.Param[0].PType := literal;
    frmMain.RPCBroker1.Param[1].Value := TIUNoteIEN;
    frmMain.RPCBroker1.Param[1].PType := literal;
    frmMain.RPCBrokerCall;
    frmMain.RPCBroker1.Call;
    if frmMain.RPCBroker1.Results.Count > 0 then
        if frmMain.RPCBroker1.Results[0] <> '1' then
            MessageDlgCAPRI('Error: Unable to link exam to progress note.', mtError, [mbOK], 0);
end;

ModalResult := mrOk; //rra 917335 flag 396.7 for update when returns to TPNCSForm.xFormOutputOKClick

{ Send exam to VVA
  * Do not send an exam to VVA unless it is an C&P Exam. //CodeCR459 JRL 7/22/13
  * Do not send exam if a cosignature is needed (CosignerButton.Caption = '')
  * CosigMode = FALSE, then always send this exam to VVA
  * CosigMode = TRUE means this exam required a signature and the cosigner
    is reviewing the exam.
  * If a cosigner is signing the exam from the popup alert, or if an exam is
    being accessed from the Tools | My Unsigned Exams menu option, this means
    the patient may not have been selected yet and the global variables
    (name/ssn) are not populated. If these variables are blank, then get
    the IEN for the patient stored in ptIENHidden.Caption, retrieve the
    name and ssn, and populate the global variables PatientName and
    PatientSSN so these values exist on the main form when the document is
    sent to VVA. Otherwise, the transmission to VVA will error. Then reset
    the variables so blank so nothing is left behind from the transmit.

    Original SendToVVA code: CodeCR423 JRL 11/13/12
    Update SendToVVA code handling cosigned exams CodeCR423 JRL 03/08/12
    Updated to handle sending to the wrong VVA folder JRL 05/03/13
      because if the same issue as the cosigned exams
      coming from the Alert Popups or the Unsigned Exams
      menu item
    Updated to only send C&P Exams to VVA JRL 07/22/13
}

if CnPExam then // CodeCR459 JRL 7/22/13
begin // Only send C&P exams to VVA
    PopulatingNameSSN := FALSE; // default
    SendExamToVVA := FALSE; // default

    SavePatientName := PatientName; // Save the current selected
patient // in CAPRI to restore later
    SavePatientSSN := PatientSSN;
    selectedPatient := TPatient.Create;

    FMGetsVVPatientInfo.FieldNumbers.Clear; // If a patient is selected
from an // alert popup or goes to the
    FMGetsVVPatientInfo.FieldNumbers.Add('01'); // name
unsigned // template screen from the
    FMGetsVVPatientInfo.FieldNumbers.Add('09'); // ssn
tools
    FMGetsVVPatientInfo.FieldNumbers.Add('02'); // gender
    FMGetsVVPatientInfo.FieldNumbers.Add('03'); // date of birth format ie, 12/24/1929
    FMGetsVVPatientInfo.FieldNumbers.Add('991.01'); // icn
    FMGetsVVPatientInfo.FieldNumbers.Add('313'); // claim number
    FMGetsVVPatientInfo.FileName := '2'; // menu, the patient exam being signed
    FMGetsVVPatientInfo.IENS := ptIENHidden.Caption; // may be different from the patient
    FMGetsVVPatientInfo.GetData; // selected from normal CAPRI use.
    if FMGetsVVPatientInfo.Results.Count > 0 then //
    begin // Get Selected patient info from
        SelectedPatientName := FMGetsVVPatientInfo.GetField('01').FMDBExternal; // the IENs stored on the TIU
form
        SelectedPatientSSN := FMGetsVVPatientInfo.GetField('09').FMDBExternal;

        selectedPatient.FirstName := Piece(SelectedPatientName, ',', 1);
        selectedPatient.MiddleName := Piece(Piece(SelectedPatientName, ',', 2), ' ', 2);
        selectedPatient.LastName := Piece(Piece(SelectedPatientName, ',', 2), ' ', 1);

        selectedPatient.Gender := FMGetsVVPatientInfo.GetField('02').FMDBInternal;

```

```

selectedPatient.Birthdate := FMGetsVVAPatientInfo.GetField('.03').FMDBExternal;
selectedPatient.Birthdate := Piece(selectedPatient.Birthdate, '/', 3) + ' ' +
    Piece(selectedPatient.Birthdate, '/', 1) + ' ' + Piece(selectedPatient.Birthdate, '/', 2);
selectedPatient.IntegrationControlNumber := FMGetsVVAPatientInfo.GetField('991.01').FMDBExternal;
selectedPatient.SocialSecurityNumber := SelectedPatientSSN;
selectedPatient.ClaimNumber := FMGetsVVAPatientInfo.GetField('.313').FMDBExternal;
end
else
begin
    SelectedPatientName := '';
    SelectedPatientSSN := '';
end;

if (PatientName = '') and (PatientSSN = '') and
    (SelectedPatientName = '') and (SelectedPatientSSN = '') then
begin
    SendExamToVVA := FALSE;
    ShowMessageCapri('This exam must be manually transmitted to Virtual VA');
end
else if (PatientName = SelectedPatientName) and (PatientSSN = SelectedPatientSSN) then
begin
    PopulatingNameSSN := FALSE;
    SendExamToVVA := TRUE;
end
else if (PatientName = '') or (PatientSSN = '') then
begin
    PopulatingNameSSN := TRUE;
    PatientName := SelectedPatientName;
    PatientSSN := SelectedPatientSSN;
    SendExamToVVA := TRUE;
end
else if (SelectedPatientName <> PatientName) and (SelectedPatientSSN <> PatientSSN) then
begin
    PopulatingNameSSN := TRUE;
    PatientName := SelectedPatientName;
    PatientSSN := SelectedPatientSSN;
    SendExamToVVA := TRUE;
end
else
begin
    SendExamToVVA := FALSE;
    ShowMessageCapri('This exam must be manually transmitted to Virtual VA');
end;

if CosignerButton.Caption <> '' then
    SendExamToVVA := FALSE;

if SendExamToVVA then
begin
    // Get Report data to automatically send info to the VVA
    TIUtoVVAProgressNotes := TStringList.Create;
    try
        CheckTIUNotes := GetTIUText(TIUNoteIEN, TIUtoVVAProgressNotes);
        if CheckTIUNotes then // Notes returned
            frmMain.actFileTransmitVirtualVAExecute(Self);
        finally
            FreeAndNil(TIUtoVVAProgressNotes);
        end;
    end;

{ if we're sending to VVA then we save to 1) VLER DAS and 2) VistA }
if SendExamToVVA and (isCosignatureRequired = False) then
begin
    if CoSigMode = True then
    begin
        LoadDBQForRendering(cosigTiuDocumentIEN);
        frmMain.NumExamTextForPDFs := pnCSForm.Exams.Count;
        SetLength(frmMain.ExamTextForPDFs, frmMain.NumExamTextForPDFs);
        for i := 0 to pnCSForm.Exams.Count - 1 do
        begin
            frmMain.ExamTextForPDFs[i] := TMemoryStream.create;
            FinalReport.SaveToStream(frmMain.ExamTextForPDFs[i]);
        end;
    end;
    // Created in Virtual VA code in case it is needed for cosignature text.
    FreeAndNil(FinalReport);

    if pnCSForm = Nil then
        MessageDlg('DBQ is not loaded; will NOT transmit to VLER DAS', mtError, [mbOk], 0)
    else
    begin

```

```

        SendDbqsToVlerAndVista(selectedPatient);
        if CoSigMode = True then
            begin
                UnloadDBQForRendering;
            end;
        end;

        if PopulatingNameSSN then
            begin
                PatientName := SavePatientName;
                PatientSSN := SavePatientSSN;
            end;
        end; // if CnPExam

        ModalResult := mrOk;
    end;

```

Modified Logic

```

//
// OK Button   Sign the exam
//
// YES/NO     this is a C&P Exam notes
//
// If the user chooses NO to "is this a C&P exam" and there is a cosignature,
// handle the same way as "NO" without a cosignature, i.e. mark the exam as
// complete. In addition, validate the cosigner data, and then delete it. This
// avoids alerts being sent to the cosigner (desired) for non C&P exams.
//
// If the user chooses NO to "is this a C&P exam", do not send that note to VVA
// or VLER/DAS web service. Only C&P exams are sent to any webservices now.
//
// Check to see if the question "Is this a C&P exam" is begin displayed.
// If so, handle normally. If it is not being displayed, then the user is
// coming from an alert/cosignature screen. Treat the exam as a C&P exam.
//
procedure TTIUSignForm.ButtonOK2Click(Sender: TObject);
var
    Appointment: String;
    TitleIEN: String;
    ErrMsg: String;
    CheckTIUNotes : Boolean;
    SendExamToVVA : Boolean;           // CodeCR423 JRL 3/8/13
    PopulatingNameSSN : Boolean;       // CodeCR423 JRL 3/8/13
    I : Integer;                      // CodeCR471 JRL 4/30/13
    ExamIENS : TStringList;           // CodeCR471 JRL 4/30/13
    RequestIEN : String;              // CodeCR471 JRL 5/08/13
    buffer : string;                  // CodeCR471 JRL 5/08/13
    SelectedPatientName, SavePatientName : String; // JRL 5/3/13
    SelectedPatientSSN, SavePatientSSN : String;   // JRL 5/3/13
    CosignYes CnPExamNO : Boolean;    // CodeCR459 JRL 7/18/13
    CnPExam : Boolean;                // CodeCR459 JRL 7/22/13
    selectedPatient: TPatient;

    {cosignature variables needed for vler das #186}
    isCosignatureRequired : Boolean;
    cosigTiuDocumentIEN: String;
    cosigCapriTemplateIEN: String;
    TransmissionErrors : Boolean; //CodeCR562 JRL 7/17/14

begin
    { is a cosignature required? }
    isCosignatureRequired := Trim(FMedit16.Text) <> '';

    if Self.Owner is TfrmTIUCosign then
        cosigTiuDocumentIEN := (Self.Owner as TfrmTIUCosign).TiuDocumentIEN;

    { identify which exam will be the parent exam ien, addresses an edge case bug }
    for i := 0 To lstExamsRequested.Items.Count - 1 do
        begin
            if lstExamsRequested.Selected[i] = true then
                begin
                    parentExamIen := Piece(lstExamsRequested.Items[i], '^', 2);
                    ParentExamTitle := TrimRight(Piece(lstExamsRequested.Items[i], '^', 1)); // JRL 09/27/13 C&P Exam Title
                    break;
                end;
            end;
        end;

    // Is the question "Is this a C&P exam" being displayed?

```

```

if cmbxCPRqst.Visible then
    CnPExam := SameText(UpperCase(cmbxCPRqst.Text), 'YES')
else
    CnPExam := TRUE;

// If the user chooses NO to "is this a C&P exam" and there is a cosignature,
// handle the same way as "NO" without a cosignature, i.e. mark the exam as
// complete.
if (CnPExam = FALSE) and
    (cosignerButton.Caption <> '') then
    CosignYes CnPExamNO := TRUE
else
    CosignYes CnPExamNO := FALSE;

ComboBoxExamLocation.ItemIndex := 0;

// Set to release report in AMIE by default.
ComboBoxReleaseReport.ItemIndex := 0;

CosigMode := False;
if CoSigTIUNoteIEN <> '' then
begin
    TIUNoteIEN := CoSigTIUNoteIEN;
    CosigMode := True;
end;

//Validate user input fields Short circuit on failure
if NOT IsUserInputValid(CosigMode) then
    Exit;

// wait until validation is done. Then check if the condition of a Non C&P
// Exam is being signed that requires a cosignature. If so, delete the
// cosignature information and send the TIU note only. Removing the cosignature
// information keeps the cosigner from getting an "alert" when they login to
// CAPRI. The cosignature process is handled outside of CAPRI for non C&P
// exams so we don't care who the cosigner was supposed to be. Bottom line is
// sacrificing storing this data to avoid the alert.
if CosignYes CnPExamNO = TRUE then
begin
    cosignerButton.Caption := '';
    FMEdit16.Text := ''; // Clear out saved IEN
end;

//Are any exams pending Short circuit if NO
if CnPExam then //CodeCR459 JRL 5/21/13
begin
    if FMEExamRequestListbox.Items.Count = 0 then
    begin
        ShowMessageCAPRI('There are no AMIE exam requests needing results. This document must be linked to an
open request before it can be uploaded. ');
        ModalResult := mrCancel;
        Exit;
    end;
end; { End If Template was for C&P Request }

if CosigMode = False then
begin {Start NOT CosigMode}
    {===== Swap date and clinic location =====}
    VisitIDIEN := ''; // clear any previous visit IENS
    if Listbox4.ItemIndex > 1 then
        VisitIDIEN := Piece(Listbox5.Items[Listbox4.ItemIndex], '^', 2) + ';'
        + Piece(Listbox5.Items[Listbox4.ItemIndex], '^', 1) + ';H'
    else if Listbox1.ItemIndex > 1 then
    begin
        Appointment := FindMatchingApptData;
        VisitIDIEN := Piece(Appointment, '^', 2) + ';'
        + Piece(Appointment, '^', 1) + ';A';
    end;

    {===== Create and Populate TIU Record =====}
    if TIUNoteIEN = '' then
    begin
        TitleIEN := GetTIUTitleIEN;
        TIUNoteIEN := CreateTIURecord(VisitIDIEN, TitleIEN);

        if Piece(TIUNoteIEN, '^', 1) = '0' then
        begin
            ShowMessageCAPRI('Could not create TIU Record. Cannot Continue. ' + Piece(TIUNoteIEN, '^', 2));
            ModalResult := mrCancel;
            Exit;
        end;
    end;
end;

```

```

// Populate TIU record text
ErrMsg := '';
if PNCSForm.xSampleReportOutput.Lines.Count > 0 then
    SetText(ErrMsg, PNCSForm.xSampleReportOutput.Lines, StrToInt(TIUNoteIEN), 0);

if ErrMsg <> '' then
    ShowMessageCAPRI(ErrMsg);
end;

PNCSForm.xPanelSave.Visible := False;
PNCSForm.xGaugel.Progress := 0;
TIUSignForm.Visible := True;

{===== Validate TIU Text transfer =====}
if NOT IsTIUTextValid(TIUNoteIEN) then
begin
    ModalResult := mrCancel;
    Exit;
end;

{==== Sign TIU Record and if failed, then try to delete TIU ====}
if SignTIURecord(TIUNoteIEN, Edit3.Text) then
begin
    //Update status and TIU Document # in CAPRI and AMIE files
    PNCSForm.xFMEdit10.Text := TIUNoteIEN; // Set TIU Document number in CAPRI template file
    // Set Review Status
    if cosignerButton.Caption <> '' then
        pnCSForm.xFMEditReviewStatus.Text := 'U'
    else
        pnCSForm.xFMEditReviewStatus.Text := 'C';
    if CosignYes CnPEXamNO then // CodeCR459 JRL 7/22/13 If cosigning a non c&p exam,
then set
        pnCSForm.xFMEditReviewStatus.Text := 'C'; // CodeCR459 JRL 7/22/13 the status to complete.
        pnCSForm.xFMEdit8.Text := 'NOW'; {Set date/time of sig to lock note in IPR file}
    end
    else
    begin
        DeleteTIURecord(TIUNoteIEN, '');
        ModalResult := mrCancel;
        Exit;
    end;
end {End NOT CosigMode}
else
begin {Start CosigMode}
    if SignTIURecord(TIUNoteIEN, Edit3.Text) then
    begin
        if frmTIUCosign.addendumText.Lines.Count > 0 then
            frmTIUCosign.SaveAddendum;
        end
        else
        Exit;
    end;
end; {End CosigMode}

// Get Episode Date/Time
anuApptPointer := ''; //CodeCR499 JRL 6/27/13
if not CheckBoxOpinion.Checked then //CodeCR499 JRL 6/27/13
    anuApptPointer := GetApptPointer(TIUNoteIEN);

if CnPEXam then //CodeCR459 JRL 7/22/13
begin //Added if statement and
encapsulated //existing code

    // Set AMIE with TIU text
    if labelCosigner.enabled = false then
        SetAMIEResults(TIUNoteIEN, VisitIDIEN, anuApptPointer);

    //Store linked AMIE exams in CAPRI Template file
    SetExamInTemplate(coSigMode);

    if labelCosigner.enabled = false then
    begin
        // Send out email for completed exam(s) // Added email send routine
        try // from try/finally inclusive
            ExamIENS := TStringList.Create; // CodeCR471 JRL 5/1/13
            ExamIENS.Clear; // Relocated code JRL 6/25/13
            for I := 0 To lstExamsRequested.Items.Count - 1 do //
            begin //
                if lstExamsRequested.Selected[I] = true then // Get selected exam IEN
                begin // and add to exam list
                    buffer := Piece(lstExamsRequested.Items[I], '^', 2); //
                end
            end
        end
    end
end

```

```

        ExamIENS.Add(buffer);
    end;
end;
RequestIEN := FMExamRequestListbox.GetSelectedRecord.IEN;
if NOT SendNotificationTwo(ExamIENS,
    AuthorIEN,
    RequestIEN) then
    ShowMessageCAPRI('There was a problem sending a completion email notification. Please notify
IRM!');
finally
    FreeAndNil(ExamIENS);
end;

FMExamRequestListboxClick(Application);
// Be sure that nothing is still open
if IsExamOKtoRelease then
    ReleaseExamToRO; // Release the results to the RO

end;
end; { End If Template was for C&P Request }

TIUSignForm.Visible := False;

if CosigMode = True then
    // Find template that matches this document in CAPRI and change
    // status to complete.
    SetMatchingTemplateComplete(TIUNoteIEN)
else if isCosignatureRequired = True then
begin
    frmMain.RPCBroker1.Results.Clear;
    frmMain.RPCBroker1.Param.Clear;
    frmMain.RPCBroker1.RemoteProcedure := 'DVBA CAPRI EXAM LINK TIU';
    frmMain.RPCBroker1.Param[0].Value := Piece(PNCSTForm.xFMEdit2.IENS, ',', 1);
    frmMain.RPCBroker1.Param[0].PType := literal;
    frmMain.RPCBroker1.Param[1].Value := TIUNoteIEN;
    frmMain.RPCBroker1.Param[1].PType := literal;
    frmMain.RPCBrokerCall;
    frmMain.RPCBroker1.Call;
    if frmMain.RPCBroker1.Results.Count > 0 then
        if frmMain.RPCBroker1.Results[0] <> '1' then
            MessageDlgCAPRI('Error: Unable to link exam to progress note.', mtError, [mbOK], 0);
        end;
end;

ModalResult := mrOk; //rra 917335 flag 396.7 for update when returns to TPNCSTForm.xFormOutputOKClick

{ Send exam to VVA
    * Do not send an exam to VVA unless it is an C&P Exam.
    * Do not send exam if a cosignature is needed (CosignerButton.Caption = '')
    * CosigMode = FALSE, then always send this exam to VVA
    * CosigMode = TRUE means this exam required a signature and the cosigner
      is reviewing the exam.
    * If a cosigner is signing the exam from the popup alert, or if an exam is
      being accessed from the Tools | My Unsigned Exams menu option, this means
      the patient may not have been selected yet and the global variables
      (name/ssn) are not populated. If these variables are blank, then get
      the IEN for the patient stored in ptIENHidden.Caption, retrieve the
      name and ssn, and populate the global variables PatientName and
      PatientSSN so these values exist on the main form when the document is
      sent to VVA. Otherwise, the transmission to VVA will error. Then reset
      the variables so blank so nothing is left behind from the transmit.

    Original SendToVVA code:
    Update SendToVVA code handling cosigned exams
    Updated to handle sending to the wrong VVA folder
    because if the same issue as the cosigned exams
    coming from the Alert Popups or the Unsigned Exams
    menu item
    Updated to only send C&P Exams to VVA

    CodeCR423 JRL 11/13/12
    CodeCR423 JRL 03/08/12
    JRL 05/03/13
    JRL 07/22/13
}

if CnPExam then
begin
    PopulatingNameSSN := FALSE;
    SendExamToVVA := FALSE;

    SavePatientName := PatientName;
    SavePatientSSN := PatientSSN;
    selectedPatient := TPatient.Create;

    FMGetsVVAPatientInfo.FieldNumbers.Clear;
    FMGetsVVAPatientInfo.FieldNumbers.Add('.01'); // name

```



```

        // alert popup or goes to the unsigned
FMGetsVVAPatientInfo.FieldNumbers.Add('.09'); // ssn                // template screen from the tools
FMGetsVVAPatientInfo.FieldNumbers.Add('.02'); // gender
FMGetsVVAPatientInfo.FieldNumbers.Add('.03'); // date of birth    format ie, 12/24/1929
FMGetsVVAPatientInfo.FieldNumbers.Add('991.01'); // icn
FMGetsVVAPatientInfo.FieldNumbers.Add('.313'); // claim number
FMGetsVVAPatientInfo.FileNumber := '2';                // menu, the patient exam being signed
FMGetsVVAPatientInfo.IENS := ptIENHidden.Caption;      // may be different from the patient
FMGetsVVAPatientInfo.GetData;                          // selected from normal CAPRI use.
if FMGetsVVAPatientInfo.Results.Count > 0 then          //
begin                                                    // Get Selected patient info from
    SelectedPatientName := FMGetsVVAPatientInfo.GetField('.01').FMDBExternal; // the IENs stored on the TIU
form
    SelectedPatientSSN := FMGetsVVAPatientInfo.GetField('.09').FMDBExternal;

    selectedPatient.FirstName := Piece(SelectedPatientName, ',', 1);
    selectedPatient.MiddleName := Piece(Piece(SelectedPatientName, ',', 2), ' ', 2);
    selectedPatient.LastName := Piece(Piece(SelectedPatientName, ',', 2), ' ', 1);

    selectedPatient.Gender := FMGetsVVAPatientInfo.GetField('.02').FMDBInternal;
    selectedPatient.Birthdate := FMGetsVVAPatientInfo.GetField('.03').FMDBExternal;
    selectedPatient.Birthdate := Piece(selectedPatient.Birthdate, '/', 3) + ' ' +
        Piece(selectedPatient.Birthdate, '/', 1) + ' ' + Piece(selectedPatient.Birthdate, '/', 2);
    selectedPatient.IntegrationControlNumber := FMGetsVVAPatientInfo.GetField('991.01').FMDBExternal;
    selectedPatient.SocialSecurityNumber := SelectedPatientSSN;
    selectedPatient.ClaimNumber := FMGetsVVAPatientInfo.GetField('.313').FMDBExternal;
end
else
begin
    SelectedPatientName := '';
    SelectedPatientSSN := '';
end;

if (PatientName = '') and (PatientSSN = '') and          //
    (SelectedPatientName = '') and (SelectedPatientSSN = '') then // We don't know who this is,
begin                                                    // show an error.
    SendExamToVVA := FALSE;
    ShowMessageCapri('This exam must be manually transmitted to Virtual VA');
end
else if (PatientName = SelectedPatientName) and (PatientSSN = SelectedPatientSSN) then //
begin                                                    // Normal mode    name on the
    PopulatingNameSSN := FALSE;                          // signature screen match name
    SendExamToVVA := TRUE;                                // signed into CAPRI
end
else if (PatientName = '') or (PatientSSN = '') then    //
begin                                                    // Probably got here from a
no                                                        // Resolve Alert button since
    PopulatingNameSSN := TRUE;                            // CAPRI patient is selected.
    PatientName := SelectedPatientName;                  // Populate the patient name
    PatientSSN := SelectedPatientSSN;                    // temporarily before sending
    SendExamToVVA := TRUE;                                // to VVA
end
else if (SelectedPatientName <> PatientName) and (SelectedPatientSSN <> PatientSSN) then //
begin                                                    // Probably got here from the
    PopulatingNameSSN := TRUE;                            // Tools Menu | Unsigned
Templates.
    PatientName := SelectedPatientName;                  // This means a CAPRI patient
is
    PatientSSN := SelectedPatientSSN;                    // selected but that is not the
    SendExamToVVA := TRUE;                                // patient being signed.
Populate
end                                                        // the patient name temporarily
for
else                                                        // sending to VVA
begin                                                    //
    SendExamToVVA := FALSE;                                // We don't know who this is,
    ShowMessageCapri('This exam must be manually transmitted to Virtual VA'); // show an error.
end;

if CosignerButton.Caption <> '' then                      // Cosignature needed, don't send
    SendExamToVVA := FALSE;                                // to VVA regardless of code above

if SendExamToVVA then                                     // CodeCR423 JRL 03/08/13
begin
    // Get Report data to autoatically send info to the VVA // CodeCR423 JRL 11/13/12
    TIUtoVVAProgressNotes := TStringList.Create;          // CodeCR423 JRL 11/13/12
    try                                                    // CodeCR423 JRL 11/13/12
        CheckTIUNotes := GetTIUText(TIUNoteIEN, TIUtoVVAProgressNotes); // CodeCR423 JRL 11/13/12
        if CheckTIUNotes then // Notes returned           // CodeCR423 JRL 11/14/12
            frmMain.actFileTransmitVirtualVAExecute(Self); // CodeCR423 JRL 11/13/12
        finally                                           // CodeCR423 JRL 11/13/12

```

```

        FreeAndNil(TIUtoVVAProgressNotes);
    end;
end;

{ if we're sending to VVA then we save to 1) VLER DAS and 2) VistA }
if SendExamToVVA and (isCosignatureRequired = False) then
begin
    if CoSigMode = True then
    begin
        LoadDBQForRendering(cosigTiuDocumentIEN);
        frmMain.NumExamTextForPDFs := pncsForm.Exams.Count;
        SetLength(frmMain.ExamTextForPDFs, frmMain.NumExamTextForPDFs);
        for i := 0 to pncsForm.Exams.Count - 1 do
        begin
            frmMain.ExamTextForPDFs[i] := TMemoryStream.create;
            FinalReport.SaveToStream(frmMain.ExamTextForPDFs[i]);
        end;
    end;
    // Created in Virtual VA code in case it is needed for cosignature text.
    FreeAndNil(FinalReport); // CodeCR565 JRL 7/20/14

    if pncsForm = Nil then
        MessageDlg('DBQ is not loaded; will NOT transmit to VLER DAS', mtError, [mbOk], 0)
    else
    begin
        // SendDbqsToVlerAndVista(selectedPatient);
        for i := 1 to pncsForm.Exams.Count do // Setup ReTransmit array // CodeCR562 JRL 7/17/14
        begin
            ExamSent[i] := FALSE; // CodeCR562 JRL 7/17/14
            SendDbqsToVlerAndVista(selectedPatient, FALSE); // Send first time // CodeCR562 JRL 7/17/14
            TrasmissionErrors := FALSE; // CodeCR562 JRL 7/17/14
            for i := 1 to pncsForm.Exams.Count do // CodeCR562 JRL 7/17/14
            begin // CodeCR562 JRL 7/17/14
                if ExamSent[i] = FALSE then // CodeCR562 JRL 7/17/14
                begin // CodeCR562 JRL 7/17/14
                    TrasmissionErrors := TRUE; // CodeCR562 JRL 7/17/14
                end; // CodeCR562 JRL 7/17/14
                if TrasmissionErrors then // CodeCR562 JRL 7/17/14
                begin // CodeCR562 JRL 7/17/14
                    if MessageDlgCAPRI('Transmission to VLER failed, do you want to
retransmit?', mtConfirmation, [mbYes, mbNo], 0) = mrYes then
                        SendDbqsToVlerAndVista(selectedPatient, TRUE); // CodeCR562 JRL 7/17/14
                    end; // else // CodeCR562 JRL 7/17/14
                end; // free memory for stored PDFs here // CodeCR562 JRL 7/18/14 moved code to accommodate
retransmission // CodeCR565 JRL 7/15/14
                for i := 0 to frmMain.NumExamTextForPDFs - 1 do // CodeCR562 JRL 7/18/14 // CodeCR565 JRL 7/15/14
                begin
                    FreeAndNil(frmMain.ExamTextForPDFs[i]); // CodeCR562 JRL 7/18/14 // CodeCR565 JRL 7/15/14
                end; // CodeCR562 JRL 7/17/14
            end;
        end;
        if CoSigMode = True then
        begin
            UnloadDBQForRendering;
        end;
    end;

    if PopulatingNameSSN then
    begin
        PatientName := SavePatientName;
        PatientSSN := SavePatientSSN;
    end;
end; // if CnPExam

ModalResult := mrOk;
end;

```

```

// if we populated the global variables
// to send the exam to VVA, restore
// them now to avoid any other issues
// in CAPRI

```

- 6.2.2.6.29. Dialog - Not applicable for CodeCR562
- 6.2.2.6.30. Help Frame - Not applicable for CodeCR562
- 6.2.2.6.31. HL7 Application Parameter - Not applicable for CodeCR562
- 6.2.2.6.32. HL7 Logical Link - Not applicable for CodeCR562
- 6.2.2.6.33. COTS Interface - Not applicable for CodeCR562
- 6.2.2.7. CODECR563 – Modify XML for In-House Exams
 - 6.2.2.7.1. Routines (Entry Points) – Not applicable for CodeCR 563
 - 6.2.2.7.2. Templates – Not applicable for CodeCR 563
 - 6.2.2.7.3. Bulletins – Not applicable for CodeCR 563
 - 6.2.2.7.4. Data Entries Affected by the Design – Not applicable for CodeCR 563
 - 6.2.2.7.5. Unique Record(s) – Not applicable for CodeCR 563
 - 6.2.2.7.6. File or Global Size Changes – Not applicable for CodeCR 563
 - 6.2.2.7.7. Mail Groups – Not applicable for CodeCR 563
 - 6.2.2.7.8. Security Keys – Not applicable for CodeCR 563
 - 6.2.2.7.9. Options – Not applicable for CodeCR 563
 - 6.2.2.7.10. Protocols – Not applicable for CodeCR 563
 - 6.2.2.7.11. Remote Procedure Call (RPC) – Not applicable for CodeCR 563
 - 6.2.2.7.12. Constants Defined in Interface – Not applicable for CodeCR 563
 - 6.2.2.7.13. Variables Defined in Interface – Not applicable for CodeCR 563
 - 6.2.2.7.14. Types Defined in Interface – Not applicable for CodeCR 563
 - 6.2.2.7.15. GUI

Table 36: GUI

Unit Name	Description
	TIUSign.pas

- 6.2.2.7.16. GUI Classes – Not applicable for CodeCR 563
- 6.2.2.7.17. Current Form – Not applicable for CodeCR 563
- 6.2.2.7.18. Modified Form – Not applicable for CodeCR 563
- 6.2.2.7.19. Components on Form – Not applicable for CodeCR 563
- 6.2.2.7.20. Events – Not applicable for CodeCR 563
- 6.2.2.7.21. Methods – Not applicable for CodeCR 563
- 6.2.2.7.22. Special References – Not applicable for CodeCR 563
- 6.2.2.7.23. Class Events – Not applicable for CodeCR 563
- 6.2.2.7.24. Class Methods – Not applicable for CodeCR 563
- 6.2.2.7.25. Class Properties – Not applicable for CodeCR 563
- 6.2.2.7.26. Uses Clause - Not applicable for CodeCR 563
- 6.2.2.7.27. Forms - Not applicable for CodeCR 563
- 6.2.2.7.28. Functions

Table 48: Functions

Function Name	SendDbqsToVlerAndVista
Short Description	Modify routine to populate the new data structures sent in the Common Data section to VLER/DAS
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	N/A

Function Name	SendDbqsToVlerAndVista
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference

Function Name	SendDbqsToVlerAndVista
Input Attribute Name and Definition	Name:N/A Definition:
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic

```

procedure TTIUSignForm.SendDbqsToVlerAndVista(selectedPatient: TPatient);
const
  WEB_SERVICE_BUNDLE = 'WebServiceBundle';
  EXTENSION_ZIP = '.zip';
  EXTENSION_UUE = '.uue';
var
  i: Integer;

  singleExam: TCPWMEexam;
  claimAttachment: TVlerDasAttachment;
  xmlEngine: TXmlEngine;

  xmlDocument: IXMLDocument;
  xmlDocumentNoAttachments: IXMLDocument;

  vlerDasClient: TFVlerDasClient;
  vlerDasThread: TVlerDasClientThread;
  vlerDasClientResult: TStringList;

  savedExamFiles: TStringList;
  tempFile: String;
  CookieMgr : TIdCookieManager;

begin
  frmMain.VlerDasToken := GetRemoteAuthenticationToken(frmMain.RPCBroker1);      // CodeCR540 JRL 12/11/13
  savedExamFiles := TStringList.Create;

  // create partial claim attachment object; remaining completed once xml engine is available
  if (PdfMemoryStream <> Nil) and (PdfMemoryStream.Size > 0) then
  begin
    claimAttachment := TVlerDasAttachment.Create;
    claimAttachment.BinaryBase64Object := claimAttachment.EncodeToBase64(PdfMemoryStream);
    claimAttachment.BinaryDescriptionText := 'text is replaced below';
    claimAttachment.BinaryFormatStandardName := 'application/pdf';
    claimAttachment.BinaryLocationURI := 'text is replaced below';
    claimAttachment.BinarySizeValue := IntToStr(PdfMemoryStream.Size);
    claimAttachment.BinaryCategoryText := 'text is replaced below';
  end;

  CookieMgr := TIdCookieManager.Create();                                     //CodeCR540 JRL 12/12/13
  for i := 0 to pncsForm.Exams.Count - 1 do
  begin
    if (frmTIUCosign <> Nil) and (frmTIUCosign.Showing) then
      frmTIUCosign.Repaint
    else if (pncsForm <> Nil) and (pncsForm.Showing) then
      pncsForm.Repaint;

    singleExam := pncsForm.Exams[i];

    if singleExam.IsXmlBasedExam = False then
      continue;

    xmlEngine := TXmlEngine.Create(singleExam);

    if claimAttachment <> Nil then
    begin
      claimAttachment.BinaryDescriptionText := xmlEngine.GetDocumentTitleText;
      claimAttachment.BinaryCategoryText := xmlEngine.GetDocumentTitleText;
    end;
  end;

```

```

        claimAttachment.BinaryLocationURI := xmlEngine.GetUniqueFilename(xmlEngine.GetDocumentTitleText, 'pdf');
        xmlEngine.AddClaimAttachment(claimAttachment);
    end;

    xmlEngine.SetPatient(selectedPatient);
    xmlDocument := xmlEngine.RenderToXml(pncsForm.xpanelBaseControl);
    xmlDocumentNoAttachments := xmlEngine.RenderToXml(pncsForm.xpanelBaseControl, False);

    if PARAM SAVE XML = True then
    begin
        xmlDocument.SaveToFile(GetCurrentDir + '\exam request ' + IntToStr(i) + '.xml');
        xmlDocumentNoAttachments.SaveToFile(GetCurrentDir + '\exam request noa ' + IntToStr(i) + '.xml');
    end;

    tempFile := xmlEngine.GetTempDirectory + 'exam request ' + IntToStr(i) + '.xml';
    if FileExists(tempFile) then
        DeleteFile(tempFile);
    xmlDocumentNoAttachments.SaveToFile(tempFile);
    savedExamFiles.Add(tempFile);

    { send payload to vler das }
    try
        vlerDasClient := TvVlerDasClient.Create(Self);
        vlerDasClient.SetDBQName(xmlEngine.GetDocumentTitleText);
        vlerDasClient.SetCookieManager(CookieMgr);

        vlerDasThread := TVlerDasClientThread.Create(vlerDasClient);

        vlerDasClientResult := TStringList.Create;
        vlerDasThread.VlerDasClientResult := vlerDasClientResult;

    try
        vlerDasThread.XmlDocument := xmlDocument;
        vlerDasThread.DocumentIdentificationID := xmlEngine.GetDocumentIdentificationID;
        vlerDasThread.Resume;
        vlerDasThread.WaitFor;

        if vlerDasThread.AnException <> Nil then
            raise vlerDasThread.AnException;
    except
        on E: Exception do
        begin
            if vlerDasClientResult.Text <> '' then
                vlerDasClientResult.Text := E.Message;
            MessageDlg('Failed to transmit the DBQ XML for ' +
                xmlEngine.GetDocumentTitleText + #10#13#10#13 + 'Reason: ' +
                E.Message, mtError, [mbOK], 0);
        end;
    end;

    if PARAM SAVE XML = True then
        vlerDasClientResult.SaveToFile(GetCurrentDir + '\exam response ' + IntToStr(i) + '.xml');

    tempFile := xmlEngine.GetTempDirectory + 'exam response ' + IntToStr(i) + '.xml';
    if FileExists(tempFile) then
        DeleteFile(tempFile);
    vlerDasClientResult.SaveToFile(tempFile);

    savedExamFiles.Add(tempFile);

    FreeAndNil(vlerDasThread);
finally
    Screen.Cursor := crDefault;
end;

FreeAndNil(vlerDasClient);
FreeAndNil(xmlEngine);
end;
FreeAndNil(CookieMgr);
//CodeCR540 JRL 12/12/13

if savedExamFiles.Count > 0 then
begin
    { bundle to include request and response status for each webservice invocation }
    MiniZip1.Zipfile := xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION ZIP;
    if FileExists(MiniZip1.Zipfile) then
        DeleteFile(MiniZip1.Zipfile);

    for i := 0 to savedExamFiles.Count - 1 do
    begin
        MiniZip1.AddToZipFile(savedExamFiles[i], ExtractFileName(savedExamFiles[i]));
    end;
end;

```

```

end;

{ uu encode the zip file }
if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE) then
    DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);

UUEncode(MiniZip1.Zipfile);

{ send the payload bundle to vista }
SendDbqsToVista(parentExamIen, xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);

{ clean up our mess }
for i := 0 to savedExamFiles.Count - 1 do
begin
    if FileExists(savedExamFiles[i]) then
        DeleteFile(savedExamFiles[i]);
end;

if PARAM_SAVE_XML = True then
begin
    if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP) then
        CopyFile(PAnsiChar(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP),
            PAnsiChar(GetCurrentDir + '\ ' + WEB_SERVICE_BUNDLE + EXTENSION_ZIP), False);
end;

if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP) then
    DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_ZIP);

if FileExists(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE) then
    DeleteFile(xmlEngine.GetTempDirectory + WEB_SERVICE_BUNDLE + EXTENSION_UUE);

end; // if savedExamFiles.Count > 0 then

if PdfMemoryStream <> Nil then
    FreeAndNil(PdfMemoryStream);

if claimAttachment <> Nil then
    FreeAndNil(claimAttachment);

if savedExamFiles <> Nil then
    FreeAndNil(savedExamFiles);
end;

```

Modified Logic (Changes are in bold)

```

procedure TTIUSignForm.SendDbqsToVlerAndVista(selectedPatient: TPatient);
const
    WEB_SERVICE_BUNDLE = 'WebServiceBundle';
    EXTENSION_ZIP = '.zip';
    EXTENSION_UUE = '.uue';
var
    i: Integer;

    singleExam: TCPWMEExam;
    claimAttachment: TVlerDasAttachment;
    xmlEngine: TXmlEngine;

    xmlDocument: IXMLDocument;
    xmlDocumentNoAttachments: IXMLDocument;

    vlerDasClient: TfVlerDasClient;
    vlerDasThread: TVlerDasClientThread;
    vlerDasClientResult: TStringList;

    savedExamFiles: TStringList;
    tempFile: String;
    CookieMgr : TIdCookieManager;

begin

```

```

    frmMain.VlerDasToken : GetRemoteAuthenticationToken(frmMain.RPCBroker1);      // CodeCR540 JRL
12/11/13
    savedExamFiles : TStringList.Create;

    // create partial claim attachment object; remaining completed once xml engine is available
    if (PdfMemoryStream <> Nil) and (PdfMemoryStream.Size > 0) then
    begin
        claimAttachment : TVlerDasAttachment.Create;
        claimAttachment.BinaryBase64Object : claimAttachment.EncodeToBase64(PdfMemoryStream);
        claimAttachment.BinaryDescriptionText : 'text is replaced below';
        claimAttachment.BinaryFormatStandardName : 'application/pdf';
        claimAttachment.BinaryLocationURI : 'text is replaced below';
        claimAttachment.BinarySizeValue : IntToStr(PdfMemoryStream.Size);
        claimAttachment.BinaryCategoryText : 'text is replaced below';
    end;

    CookieMgr : TIdCookieManager.Create();                                     //CodeCR540 JRL
12/12/13
    for i : 0 to pncsForm.Exams.Count - 1 do
    begin
        if (frmTIUCosign <> Nil) and (frmTIUCosign.Showing) then
            frmTIUCosign.Repaint
        else if (pncsForm <> Nil) and (pncsForm.Showing) then
            pncsForm.Repaint;

        singleExam : pncsForm.Exams[i];

        if singleExam.IsXmlBasedExam False then
            continue;

        xmlEngine : TXmlEngine.Create(singleExam);

        if claimAttachment <> Nil then
        begin
            claimAttachment.BinaryDescriptionText : xmlEngine.GetDocumentTitleText;
            claimAttachment.BinaryCategoryText : xmlEngine.GetDocumentTitleText;
            claimAttachment.BinaryLocationURI :
xmlEngine.GetUniqueFilename(xmlEngine.GetDocumentTitleText, 'pdf');
            xmlEngine.AddClaimAttachment(claimAttachment);
        end;

        xmlEngine.SetPatient(selectedPatient);
        xmlEngine.PopulateApproverInfo('','','','','');
// CodeCR563 JRL 6/13/14

xmlEngine.PopulateExamInfo('VHA CAPRI','','Inhouse','',xmlEngine.GetW3cDateTime(),'Completed');
// CodeCR563 JRL 6/13/14

        xmlDocument : xmlEngine.RenderToXml(pncsForm.xpanelBaseControl);
        xmlDocumentNoAttachments : xmlEngine.RenderToXml(pncsForm.xpanelBaseControl, False);

        if PARAM SAVE XML True then
        begin
            xmlDocument.SaveToFile(GetCurrentDir + '\exam-request-' + IntToStr(i) + '.xml');
            xmlDocumentNoAttachments.SaveToFile(GetCurrentDir + '\exam-request-noa-' + IntToStr(i) +
'.xml');
        end;

        tempFile : xmlEngine.GetTempDirectory + 'exam-request-' + IntToStr(i) + '.xml';
        if FileExists(tempFile) then
            DeleteFile(tempFile);
        xmlDocumentNoAttachments.SaveToFile(tempFile);
        savedExamFiles.Add(tempFile);

        { send payload to vler das }
        try
            vlerDasClient : TfVlerDasClient.Create(Self);
            vlerDasClient.SetDBQName(xmlEngine.GetDocumentTitleText);
            vlerDasClient.SetCookieManager(CookieMgr);

            vlerDasThread : TVlerDasClientThread.Create(vlerDasClient);

```



```

vlerDasClientResult : TStringList.Create;
vlerDasThread.VlerDasClientResult : vlerDasClientResult;

try
  vlerDasThread.XmlDocument : xmlDocument;
  vlerDasThread.DocumentIdentificationID : xmlEngine.GetDocumentIdentificationID;
  vlerDasThread.Resume;
  vlerDasThread.WaitFor;

  if vlerDasThread.AnException <> Nil then
    raise vlerDasThread.AnException;
except
  on E: Exception do
  begin
    if vlerDasClientResult.Text <> '' then
      vlerDasClientResult.Text : E.Message;
    MessageDlg('Failed to transmit the DBQ XML for ' +
      xmlEngine.GetDocumentTitleText + #10#13#10#13 + 'Reason: ' +
      E.Message, mtError, [mbOK], 0);
    end;
  end;

  if PARAM SAVE XML True then
    vlerDasClientResult.SaveToFile(GetCurrentDir + '\exam-response-' + IntToStr(i) + '.xml');

  tempFile : xmlEngine.GetTempDirectory + 'exam-response-' + IntToStr(i) + '.xml';
  if FileExists(tempFile) then
    DeleteFile(tempFile);
  vlerDasClientResult.SaveToFile(tempFile);

  savedExamFiles.Add(tempFile);

  FreeAndNil(vlerDasThread);
finally
  Screen.Cursor : crDefault;
end;

FreeAndNil(vlerDasClient);
FreeAndNil(xmlEngine);
end;
FreeAndNil(CookieMgr);
12/12/13 //CodeCR540 JRL

if savedExamFiles.Count > 0 then
begin
  { bundle to include request and response status for each webservice invocation }
  MiniZip1.Zipfile : xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION ZIP;
  if FileExists(MiniZip1.Zipfile) then
    DeleteFile(MiniZip1.Zipfile);

  for i : 0 to savedExamFiles.Count - 1 do
  begin
    MiniZip1.AddToZipFile(savedExamFiles[i], ExtractFileName(savedExamFiles[i]));
  end;

  { uu encode the zip file }
  if FileExists(xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION UUE) then
    DeleteFile(xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION UUE);

  UUEncode(MiniZip1.Zipfile);

  { send the payload bundle to vista }
  SendDbqsToVista(parentExamIen, xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE +
EXTENSION UUE);

  { clean up our mess }
  for i : 0 to savedExamFiles.Count - 1 do
  begin

```

```

        if FileExists(savedExamFiles[i]) then
            DeleteFile(savedExamFiles[i]);
        end;

        if PARAM SAVE XML    True then
        begin
            if FileExists(xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION ZIP) then
                CopyFile(PAnsiChar(xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION ZIP),
                    PAnsiChar(GetCurrentDir + '\' + WEB SERVICE BUNDLE + EXTENSION ZIP), False);
            end;

            if FileExists(xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION ZIP) then
                DeleteFile(xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION ZIP);

            if FileExists(xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION UUE) then
                DeleteFile(xmlEngine.GetTempDirectory + WEB SERVICE BUNDLE + EXTENSION UUE);

        end; // if savedExamFiles.Count > 0 then

        if PdfMemoryStream <> Nil then
            FreeAndNil(PdfMemoryStream);

        if claimAttachment <> Nil then
            FreeAndNil(claimAttachment);

        if savedExamFiles <> Nil then
            FreeAndNil(savedExamFiles);
        end;

```

6.2.2.7.29. Dialog – Not applicable for CodeCR 563

6.2.2.7.30. Help Frame – Not applicable for CodeCR 563

6.2.2.7.31. HL7 Application Parameter – Not applicable for CodeCR 563

6.2.2.7.32. HL7 Logical Link – Not applicable for CodeCR 563

6.2.2.7.33. COTS Interface – Not applicable for CodeCR 563

6.2.2.7.34. GUI

Table 36: GUI

Unit Name	Description
	VlerDasClaim.pas

- 6.2.2.7.35. GUI Classes – Not applicable for CodeCR 563
- 6.2.2.7.36. Current Form – Not applicable for CodeCR 563
- 6.2.2.7.37. Modified Form – Not applicable for CodeCR 563
- 6.2.2.7.38. Components on Form – Not applicable for CodeCR 563
- 6.2.2.7.39. Events – Not applicable for CodeCR 563
- 6.2.2.7.40. Methods – Not applicable for CodeCR 563
- 6.2.2.7.41. Special References – Not applicable for CodeCR 563
- 6.2.2.7.42. Class Events – Not applicable for CodeCR 563
- 6.2.2.7.43. Class Methods – Not applicable for CodeCR 563
- 6.2.2.7.44. Class Properties – Not applicable for CodeCR 563
- 6.2.2.7.45. Uses Clause - Not applicable for CodeCR 563
- 6.2.2.7.46. Forms - Not applicable for CodeCR 563
- 6.2.2.7.47. Functions

Table 48: Functions

Function Name	ToXML
Short Description	Modify code to create XML add in new Exam Detail and Approver detail fields.
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
N/A	N/A	N/A

Function Name	ToXML
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference

Function Name	ToXML
Input Attribute Name and Definition	Name: N/A Definition:
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic

```

function TVlerDasClaim.ToXml: IXMLDocument;
const
  NS_CLD : string = 'http://niem.gov/niem/niem_core/2.0';
  NS_NC : string = 'http://niem.gov/niem/structures/2.0';
  NS_VLER : string = 'http://niem.gov/niem/proxy/xsd/2.0';
  NS_NIEM_XSD : string = 'http://niem.gov/niem/structures/2.0';
  NS_S : string = 'http://niem.gov/niem/structures/2.0';
  NS_XSI : string = 'http://www.w3.org/2001/XMLSchema instance';
var
  xml: IXMLDocument;

  xmlNode: IXMLNode;

  commonData: IXMLNode;
  medicalData: IXMLNode;
  attachments: IXMLNode;

  i: Integer;
  anAttachment: TVlerDasAttachment;
begin
  xml := TXMLDocument.Create(nil);
  xml.Active := True;
  xml.Version := '1.0';
  xml.Encoding := 'UTF 8';
  xml.DocumentElement := xml.CreateNode('cld:Claim', ntElement, NS_CLD);
  xml.DocumentElement.DeclareNamespace('nc', NS_NC);
  xml.DocumentElement.DeclareNamespace('vler', NS_VLER);
  xml.DocumentElement.DeclareNamespace('niem xsd', NS_NIEM_XSD);
  xml.DocumentElement.DeclareNamespace('s', NS_S);
  xml.DocumentElement.DeclareNamespace('xsi', NS_XSI);
  if Length(NamespacePrefix) > 0 then
    xml.DocumentElement.DeclareNamespace(NamespacePrefix, NamespaceURI);

  xml.DocumentElement.AddChild('DocumentTypeVersion').NodeValue := DocumentTypeVersion;
  xml.DocumentElement.AddChild('ClaimID').NodeValue := ClaimID;

  commonData := xml.DocumentElement.AddChild('CommonData');
  xmlNode := commonData.AddChild('nc:Document');
  xmlNode.AddChild('nc:DocumentCreationDate').AddChild('nc:DateTime').NodeValue := DocumentCreationDate;
  xmlNode.AddChild('nc:DocumentDescriptionText').NodeValue := DocumentTitleText;
  xmlNode.AddChild('nc:DocumentIdentification').AddChild('nc:IdentificationID').NodeValue := DocumentIdentificationID;
  xmlNode.AddChild('nc:DocumentStatus').AddChild('nc:StatusText').NodeValue := 'Completed';
  xmlNode.AddChild('nc:DocumentTitleText').NodeValue := DocumentTitleText;

  xmlNode := commonData.AddChild('vler:Client');
  xmlNode.Attributes['s:id'] := 'client';
  with xmlNode.AddChild('vler:ClientIdentifier') do
  begin
    AddChild('nc:IdentificationID').NodeValue := ClientIdentificationID;
    AddChild('vler:AssigningAuthority').NodeValue := ClientAssigningAuthority;
  end;
  xmlNode.AddChild('vler:ClientStatus').AddChild('vler:ClientStatusText').NodeValue := 'OK';
  xmlNode.AddChild('nc:RoleOfPersonReference').Attributes['s:ref'] := 'one';

  xmlNode := commonData.AddChild('vler:ServiceProvider');
  xmlNode.Attributes['s:id'] := 'service_provider';
  with xmlNode.AddChild('vler:PersonIdentifier') do
  begin
    AddChild('nc:IdentificationID').NodeValue := ServiceProviderIdentificationID;
    AddChild('vler:AssigningAuthority').NodeValue := ServiceProviderAssigningAuthority;
    AddChild('nc:IdentificationJurisdictionText').NodeValue := 'CAPRI User IEN';
  end;
  xmlNode.AddChild('vler:ServiceProviderRoleTitle').NodeValue := 'Physician';
  xmlNode.AddChild('nc:RoleOfPersonReference').Attributes['s:ref'] := 'two';

  xmlNode := commonData.AddChild('nc:Person');
  xmlNode.Attributes['s:id'] := 'two';
  with xmlNode.AddChild('nc:PersonName') do
  begin
    AddChild('nc:PersonGivenName').NodeValue := ProviderFirstName;
    AddChild('nc:PersonMiddleName').NodeValue := ProviderMiddleName;
    AddChild('nc:PersonSurName').NodeValue := ProviderLastName;
  end;
end;

```

```

end;

xmlNode := commonData.AddChild('nc:Person');
xmlNode.Attributes['s:id'] := 'one';
xmlNode.AddChild('nc:PersonBirthDate').AddChild('nc:Date').NodeValue := PatientDateOfBirth;
with xmlNode.AddChild('nc:PersonName') do
begin
  AddChild('nc:PersonGivenName').NodeValue := PatientFirstName;
  AddChild('nc:PersonMiddleName').NodeValue := PatientMiddleName;
  AddChild('nc:PersonSurName').NodeValue := PatientLastName;
end;
xmlNode.AddChild('nc:PersonSexCode').NodeValue := PatientGender;
xmlNode.AddChild('nc:PersonSSNIdentification').AddChild('nc:IdentificationID').NodeValue := PatientSSN;

xmlNode := commonData.AddChild('nc:Facility');
with xmlNode.AddChild('nc:FacilityIdentification') do
begin
  AddChild('nc:IdentificationID').NodeValue := FacilityIdentificationID;
  AddChild('nc:IdentificationJurisdictionText').NodeValue := FacilityIdentificationText;
end;
xmlNode.AddChild('nc:FacilityName').NodeValue := FacilityName;

with commonData.AddChild('vler:ServiceProviderClientAssociation') do
begin
  AddChild('vler:ServiceProviderReference').Attributes['s:ref'] := 'service_provider';
  AddChild('vler:ClientReference').Attributes['s:ref'] := 'client';
end;

medicalData := xml.DocumentElement.AddChild('MedicalData');
RenderMedicalDataXml(medicalData);

attachments := xml.DocumentElement.AddChild('Attachments');

if (AttachmentsList <> Nil) and (AttachmentsList.Count > 0) then
begin
  for i := 0 to AttachmentsList.Count - 1 do
  begin
    anAttachment := AttachmentsList[i];
    with attachments.AddChild('nc:Attachment') do
    begin
      AddChild('nc:BinaryBase64Object').NodeValue := anAttachment.BinaryBase64Object;
      AddChild('nc:BinaryDescriptionText').NodeValue := anAttachment.BinaryDescriptionText;
      AddChild('nc:BinaryFormatStandardName').NodeValue := anAttachment.BinaryFormatStandardName;
      AddChild('nc:BinaryLocationURI').NodeValue := anAttachment.BinaryLocationURI;
      AddChild('nc:BinarySizeValue').NodeValue := anAttachment.BinarySizeValue;
      AddChild('nc:BinaryCategoryText').NodeValue := anAttachment.BinaryCategoryText;
    end;
  end;
end;

Result := FixDelphiXmlBugs(xml);
end;

```

Modified Logic (Changes are in bold)

```

function TVlerDasClaim.ToXml: IXMLDocument;
const
  NS CLD = 'http://niem.gov/niem/niem core/2.0';
  NS NC = 'http://niem.gov/niem/niem core/2.0';
  NS VLER = 'http://niem.gov/niem/niem core/2.0';
  NS NIEM XSD = 'http://niem.gov/niem/proxy/xsd/2.0';
  NS S = 'http://niem.gov/niem/structures/2.0';
  NS XSI = 'http://www.w3.org/2001/XMLSchema instance';
var
  xml: IXMLDocument;

  xmlNode: IXMLNode;

  commonData: IXMLNode;
  medicalData: IXMLNode;
  attachments: IXMLNode;

  i: Integer;
  anAttachment: TVlerDasAttachment;
begin
  xml := TXMLDocument.Create(Nil);
  xml.Active := True;
  xml.Version := '1.0';
  xml.Encoding := 'UTF 8';
  xml.DocumentElement := xml.CreateNode('cld:Claim', ntElement, NS CLD);
  xml.DocumentElement.DeclareNamespace('nc', NS NC);
  xml.DocumentElement.DeclareNamespace('vler', NS VLER);
  xml.DocumentElement.DeclareNamespace('niem xsd', NS NIEM XSD);
  xml.DocumentElement.DeclareNamespace('s', NS S);
  xml.DocumentElement.DeclareNamespace('xsi', NS XSI);
  if Length(NamespacePrefix) > 0 then

```

```

xml.DocumentElement.DeclareNamespace(NamespacePrefix, NamespaceURI);

xml.DocumentElement.AddChild('DocumentTypeVersion').NodeValue := DocumentTypeVersion;
xml.DocumentElement.AddChild('ClaimID').NodeValue := ClaimID;

commonData := xml.DocumentElement.AddChild('CommonData');
xmlNode := commonData.AddChild('nc:Document');
xmlNode.AddChild('nc:DocumentCreationDate').AddChild('nc:DateTime').NodeValue := DocumentCreationDate;
xmlNode.AddChild('nc:DocumentDescriptionText').NodeValue := DocumentTitleText;
xmlNode.AddChild('nc:DocumentIdentification').AddChild('nc:IdentificationID').NodeValue :=
DocumentIdentificationID;
xmlNode.AddChild('nc:DocumentStatus').AddChild('nc:StatusText').NodeValue := StatusText; // Removed
hardcoding of "Completed" CodeCR563 JRL 6/13/14
xmlNode.AddChild('nc:DocumentTitleText').NodeValue := DocumentTitleText;

xmlNode := commonData.AddChild('vler:Client');
xmlNode.Attributes['s:id'] := 'client';
with xmlNode.AddChild('vler:ClientIdentifier') do
begin
AddChild('nc:IdentificationID').NodeValue := ClientIdentificationID;
AddChild('vler:AssigningAuthority').NodeValue := ClientAssigningAuthority;
end;
xmlNode.AddChild('vler:ClientStatus').AddChild('vler:ClientStatusText').NodeValue := 'OK';
xmlNode.AddChild('nc:RoleOfPersonReference').Attributes['s:ref'] := 'one';

xmlNode := commonData.AddChild('vler:ServiceProvider');
xmlNode.Attributes['s:id'] := 'service provider';
with xmlNode.AddChild('vler:PersonIdentifier') do
begin
AddChild('nc:IdentificationID').NodeValue := ServiceProviderIdentificationID;
AddChild('vler:AssigningAuthority').NodeValue := ServiceProviderAssigningAuthority;
AddChild('nc:IdentificationJurisdictionText').NodeValue := 'CAPRI User IEN';
end;
xmlNode.AddChild('vler:ServiceProviderRoleTitle').NodeValue := 'Physician';
xmlNode.AddChild('nc:RoleOfPersonReference').Attributes['s:ref'] := 'two';

xmlNode := commonData.AddChild('nc:Person');
xmlNode.Attributes['s:id'] := 'two';
with xmlNode.AddChild('nc:PersonName') do
begin
AddChild('nc:PersonGivenName').NodeValue := ProviderFirstName;
AddChild('nc:PersonMiddleName').NodeValue := ProviderMiddleName;
AddChild('nc:PersonSurName').NodeValue := ProviderLastName;
end;

xmlNode := commonData.AddChild('nc:Person');
xmlNode.Attributes['s:id'] := 'one';
xmlNode.AddChild('nc:PersonBirthDate').AddChild('nc:Date').NodeValue := PatientDateOfBirth;
with xmlNode.AddChild('nc:PersonName') do
begin
AddChild('nc:PersonGivenName').NodeValue := PatientFirstName;
AddChild('nc:PersonMiddleName').NodeValue := PatientMiddleName;
AddChild('nc:PersonSurName').NodeValue := PatientLastName;
end;
xmlNode.AddChild('nc:PersonSexCode').NodeValue := PatientGender;
xmlNode.AddChild('nc:PersonSSNIdentification').AddChild('nc:IdentificationID').NodeValue := PatientSSN;

xmlNode := commonData.AddChild('nc:Facility');
with xmlNode.AddChild('nc:FacilityIdentification') do
begin
AddChild('nc:IdentificationID').NodeValue := FacilityIdentificationID;
AddChild('nc:IdentificationJurisdictionText').NodeValue := FacilityIdentificationText;
end;
xmlNode.AddChild('nc:FacilityName').NodeValue := FacilityName;

with commonData.AddChild('vler:ServiceProviderClientAssociation') do
begin
AddChild('vler:ServiceProviderReference').Attributes['s:ref'] := 'service provider';
AddChild('vler:ClientReference').Attributes['s:ref'] := 'client';
end;

// add in approver detail
xmlNode := commonData.AddChild('vler:Approver'); // CodeCR563 JRL 6/9/14
xmlNode.Attributes['s:id'] := 'approver'; // CodeCR563 JRL 6/9/14
xmlNode.AddChild('PersonIdentifier').NodeValue := ''; // CodeCR563 JRL 6/9/14
xmlNode.AddChild('vler:ServiceProviderRoleTitle').NodeValue := 'EXAMINER'; // CodeCR563 JRL 6/9/14
xmlNode.AddChild('nc:RoleOfPersonReference').Attributes['s:ref'] := 'three'; // CodeCR563 JRL 6/9/14
xmlNode := commonData.AddChild('nc:Person'); // CodeCR563 JRL 6/9/14
xmlNode.Attributes['s:id'] := 'three'; // CodeCR563 JRL 6/9/14
with xmlNode.AddChild('nc:PersonName') do // CodeCR563 JRL 6/9/14
begin // CodeCR563 JRL 6/9/14

```

```

    AddChild('nc:PersonGivenName').NodeValue := ApproverFirstName;           // CodeCR563 JRL 6/9/14
    AddChild('nc:PersonMiddleName').NodeValue := ApproverMiddleName;         // CodeCR563 JRL 6/9/14
    AddChild('nc:PersonSurName').NodeValue := ApproverLastName;              // CodeCR563 JRL 6/9/14
end;                                                                           // CodeCR563 JRL 6/9/14

// add in exam detail
xmlNode := commonData.AddChild('vler:ExamDetail');                           // CodeCR563 JRL 6/9/14
xmlNode.AddChild('vler:PhysicalSource').NodeValue := PhysicalSource;        // CodeCR563 JRL 6/9/14
xmlNode.AddChild('vler:ExamId').NodeValue := ExamID;                        // CodeCR563 JRL 6/9/14
xmlNode.AddChild('vler:ExaminerType').NodeValue := ExaminerType;           // CodeCR563 JRL 6/9/14
xmlNode.AddChild('vler:ContractType').NodeValue := ContractType;           // CodeCR563 JRL 6/9/14
xmlNode.AddChild('vler:StatusDate').NodeValue := StatusDate;               // CodeCR563 JRL 6/9/14
xmlNode.AddChild('vler:Comments').NodeValue := Comments;                   // CodeCR563 JRL 6/9/14

medicalData := xml.DocumentElement.AddChild('MedicalData');
RenderMedicalDataXml(medicalData);

attachments := xml.DocumentElement.AddChild('Attachments');

if (AttachmentsList <> Nil) and (AttachmentsList.Count > 0) then
begin
    for i := 0 to AttachmentsList.Count - 1 do
    begin
        anAttachment := AttachmentsList[i];
        with attachments.AddChild('nc:Attachment') do
        begin
            AddChild('nc:BinaryBase64Object').NodeValue := anAttachment.BinaryBase64Object;
            AddChild('nc:BinaryDescriptionText').NodeValue := anAttachment.BinaryDescriptionText;
            AddChild('nc:BinaryFormatStandardName').NodeValue := anAttachment.BinaryFormatStandardName;
            AddChild('nc:BinaryLocationURI').NodeValue := anAttachment.BinaryLocationURI;
            AddChild('nc:BinarySizeValue').NodeValue := anAttachment.BinarySizeValue;
            AddChild('nc:BinaryCategoryText').NodeValue := anAttachment.BinaryCategoryText;
        end;
    end;
end;

Result := FixDelphiXmlBugs(xml);
end;

```

6.2.2.7.48. Dialog – Not applicable for CodeCR 563

6.2.2.7.49. Help Frame – Not applicable for CodeCR 563

6.2.2.7.50. HL7 Application Parameter – Not applicable for CodeCR 563

6.2.2.7.51. HL7 Logical Link – Not applicable for CodeCR 563

6.2.2.7.52. COTS Interface – Not applicable for CodeCR 563

6.2.2.7.53. GUI

Table 36: GUI

Unit Name	Description
	XMLEngine.pas

- 6.2.2.7.54. GUI Classes – Not applicable for CodeCR 563
- 6.2.2.7.55. Current Form – Not applicable for CodeCR 563
- 6.2.2.7.56. Modified Form – Not applicable for CodeCR 563
- 6.2.2.7.57. Components on Form – Not applicable for CodeCR 563
- 6.2.2.7.58. Events – Not applicable for CodeCR 563
- 6.2.2.7.59. Methods – Not applicable for CodeCR 563
- 6.2.2.7.60. Special References – Not applicable for CodeCR 563
- 6.2.2.7.61. Class Events – Not applicable for CodeCR 563
- 6.2.2.7.62. Class Methods – Not applicable for CodeCR 563
- 6.2.2.7.63. Class Properties – Not applicable for CodeCR 563
- 6.2.2.7.64. Uses Clause – Not applicable for CodeCR 563
- 6.2.2.7.65. Forms – Not applicable for CodeCR 563
- 6.2.2.7.66. Functions

Table 48: Functions

Function Name	RenderToXML
Short Description	Added additional field population for Approver and Exam Detail.
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
N/A	N/A	N/A

Function Name	RenderToXML
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference

Function Name	RenderToXML
Input Attribute Name and Definition	Name: N/A Definition:
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic

```
function TXmlEngine.RenderToXml(rootPanel: TPanel; includeAttachments: Boolean true): IXMLDocument;
var
  XusGetUserInfoResult: TStrings;
begin
  VlerDasClaim.SetRootPanel(rootPanel);

  VlerDasClaim.SetXmlNamespace(XmlConfig.GetValue(XmlConfig.NAMESPACE_PREFIX),
    XmlConfig.GetValue(XmlConfig.NAMESPACE));

  if includeAttachments True then
    VlerDasClaim.SetAttachments(AttachmentsList)
  else
    VlerDasClaim.SetAttachments(nil);

  VlerDasClaim.DocumentTypeVersion : XmlConfig.GetValue(TXmlConfig.VERSION);
  VlerDasClaim.DocumentVersion : XmlConfig.GetValue(TXmlConfig.VERSION);
  VlerDasClaim.DocumentTitleText : XmlConfig.GetValue(TXmlConfig.DOCUMENT_TITLE_TEXT);

  DocumentIdentificationID : GetStringGUID; /** web service invocation also needs DocumentIdentificationID */
  VlerDasClaim.DocumentIdentificationID : DocumentIdentificationID;
  VlerDasClaim.DocumentCreationDate : GetW3cDateTime;

  VlerDasClaim.ClaimID : Patient.SocialSecurityNumber;

  if Length(Patient.IntegrationControlNumber) > 0 then
    begin
      VlerDasClaim.ClientIdentificationID : Patient.IntegrationControlNumber;
      VlerDasClaim.ClientAssigningAuthority : VlerDasClaim.CLIENT_ASSIGNING_AUTHORITY_VA;
    end
  else
    begin
      VlerDasClaim.ClientIdentificationID : Patient.SocialSecurityNumber;
      VlerDasClaim.ClientAssigningAuthority : VlerDasClaim.CLIENT_ASSIGNING_AUTHORITY_SSA;
    end;

  XusGetUserInfoResult : GetXusUserInfo; /** calls RPC: XUS GET USER INFO */

  VlerDasClaim.ServiceProviderIdentificationID : XusGetUserInfoResult[0];
  VlerDasClaim.ServiceProviderAssigningAuthority : VlerDasClaim.CLIENT_ASSIGNING_AUTHORITY_VA;
  VlerDasClaim.ServiceProviderRoleTitle : XusGetUserInfoResult[4];

  VlerDasClaim.PatientFirstName : Patient.FirstName;
  VlerDasClaim.PatientMiddleName : Patient.MiddleName;
  VlerDasClaim.PatientLastName : Patient.LastName;
  VlerDasClaim.PatientGender : Patient.Gender;
  VlerDasClaim.PatientSSN : Patient.SocialSecurityNumber;
  VlerDasClaim.PatientDateOfBirth : Patient.Birthdate;

  VlerDasClaim.ProviderFirstName : Piece(XusGetUserInfoResult[1], ',', 2);
  VlerDasClaim.ProviderMiddleName : Piece(Piece(XusGetUserInfoResult[1], ',', 2), ' ', 2);
  VlerDasClaim.ProviderLastName : Piece(XusGetUserInfoResult[1], ',');

  VlerDasClaim.FacilityIdentificationID : Piece(XusGetUserInfoResult[3], '^', 3);
  VlerDasClaim.FacilityIdentificationText : Uppercase(frmMain.RPCBroker1.ANUStrServer); //Piece(XusGetUserInfoResult[3], '^',
2);
  VlerDasClaim.FacilityName : Uppercase(frmMain.RPCBroker1.ANUStrServer); //Piece(XusGetUserInfoResult[3], '^', 2);

  Result : VlerDasClaim.ToXml;
end;
```

Modified Logic (Changes are in bold)

```

function TXmlEngine.RenderToXml(rootPanel: TPanel; includeAttachments: Boolean = true): IXMLDocument;
var
    XusGetUserInfoResult: TStrings;
begin
    VlerDasClaim.SetRootPanel(rootPanel);

    VlerDasClaim.SetXmlNamespace(XmlConfig.GetValue(XmlConfig.NAMESPACE_PREFIX),
        XmlConfig.GetValue(XmlConfig.NAMESPACE));

    if includeAttachments = True then
        VlerDasClaim.SetAttachments(AttachmentsList)
    else
        VlerDasClaim.SetAttachments(Nil);

    VlerDasClaim.DocumentTypeVersion := XmlConfig.GetValue(TXmlConfig.VERSION);
    VlerDasClaim.DocumentVersion := XmlConfig.GetValue(TXmlConfig.VERSION);
    VlerDasClaim.DocumentTitleText := XmlConfig.GetValue(TXmlConfig.DOCUMENT_TITLE_TEXT);

    DocumentIdentificationID := GetStringGUID; /* web service invocation also needs DocumentIdentificationID
*/
    VlerDasClaim.DocumentIdentificationID := DocumentIdentificationID;
    VlerDasClaim.DocumentCreationDate := GetW3cDateTime;

    VlerDasClaim.ClaimID := Patient.SocialSecurityNumber;

    if Length(Patient.IntegrationControlNumber) > 0 then
    begin
        VlerDasClaim.ClientIdentificationID := Patient.IntegrationControlNumber;
        VlerDasClaim.ClientAssigningAuthority := VlerDasClaim.CLIENT ASSIGNING AUTHORITY VA;
    end
    else
    begin
        VlerDasClaim.ClientIdentificationID := Patient.SocialSecurityNumber;
        VlerDasClaim.ClientAssigningAuthority := VlerDasClaim.CLIENT ASSIGNING AUTHORITY SSA;
    end;

    XusGetUserInfoResult := GetXusUserInfo; /* calls RPC: XUS GET USER INFO */

    VlerDasClaim.ServiceProviderIdentificationID := XusGetUserInfoResult[0];
    VlerDasClaim.ServiceProviderAssigningAuthority := VlerDasClaim.CLIENT ASSIGNING AUTHORITY VA;
    VlerDasClaim.ServiceProviderRoleTitle := XusGetUserInfoResult[4];

    VlerDasClaim.PatientFirstName := Patient.FirstName;
    VlerDasClaim.PatientMiddleName := Patient.MiddleName;
    VlerDasClaim.PatientLastName := Patient.LastName;
    VlerDasClaim.PatientGender := Patient.Gender;
    VlerDasClaim.PatientSSN := Patient.SocialSecurityNumber;
    VlerDasClaim.PatientDateOfBirth := Patient.Birthdate;

    VlerDasClaim.ProviderFirstName := Piece(XusGetUserInfoResult[1], ',', 2);
    VlerDasClaim.ProviderMiddleName := Piece(Piece(XusGetUserInfoResult[1], ',', 2), ' ', 2);
    VlerDasClaim.ProviderLastName := Piece(XusGetUserInfoResult[1], '^', 2);

    VlerDasClaim.FacilityIdentificationID := Piece(XusGetUserInfoResult[3], '^', 3);
    VlerDasClaim.FacilityIdentificationText := Uppercase(frmMain.RPCBroker1.ANUStrServer);
//Piece(XusGetUserInfoResult[3], '^', 2);
    VlerDasClaim.FacilityName := Uppercase(frmMain.RPCBroker1.ANUStrServer); //Piece(XusGetUserInfoResult[3],
 '^', 2);

    VlerDasClaim.ApproverID := Approver.ApproverID; // CodeCR563 JRL 6/10/14
    VlerDasClaim.ApproverAssigningAuthority := Approver.ApproverAssigningAuthority; // CodeCR563 JRL 6/10/14
    VlerDasClaim.ApproverLastName := Approver.ApproverLastName; // CodeCR563 JRL 6/10/14
    VlerDasClaim.ApproverFirstName := Approver.ApproverFirstName; // CodeCR563 JRL 6/10/14
    VlerDasClaim.ApproverMiddleName := Approver.ApproverMiddleName; // CodeCR563 JRL 6/10/14

    -
    VlerDasClaim.ExamID := ExamInfo.ExamID; // CodeCR563 JRL 6/10/14
    VlerDasClaim.PhysicalSource := ExamInfo.PhysicalSource; // CodeCR563 JRL 6/10/14
    VlerDasClaim.ExaminerType := ExamInfo.ExaminerType; // CodeCR563 JRL 6/10/14
    VlerDasClaim.ContractType := ExamInfo.ContractType; // CodeCR563 JRL 6/10/14
    VlerDasClaim.StatusDate := GetW3cDateTime; // CodeCR563 JRL 6/10/14
    VlerDasClaim.Comments := ExamInfo.Comments; // CodeCR563 JRL 6/10/14
    VlerDasClaim.StatusText := ExamInfo.StatusText; // CodeCR563 JRL 6/10/14

    Result := VlerDasClaim.ToXml;
end;

```

Function Name	PopulateExamInfo
Short Description	Added additional field population for Approver and Exam Detail.
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	N/A

Function Name	PopulateExamInfo
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name:N/A Definition:
Output Attribute Name and Definition	Name:N/A Definition:

Current Logic
N/A

Modified Logic (Changes are in bold)

Modified Logic (Changes are in bold)

```

{-----
  Added routine to allow population of these fields from TIUSign
  CodeCR563 JRL 6/13/14
-----}
procedure TXmlEngine.PopulateExamInfo(ExamPhysSrc, ExamID, ExamType, ConType, StatDt,
Comments, StatusTxt : String);
begin
  ExamInfo.PhysicalSource := ExamPhysSrc;
  ExamInfo.ExamID := ExamID;
  ExamInfo.ExaminerType := ExamType;
  ExamInfo.ContractType := ConType;
  ExamInfo.StatusDate := StatDt;
  ExamInfo.Comments := Comments;
  ExamInfo.StatusText := StatusTxt;
end;

```

Function Name	PopulateApproverInfo
Short Description	Added additional field population for Approver and Exam Detail.
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
N/A	N/A	N/A

Function Name	PopulateApproverInfo
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name:N/A Definition:

Function Name	PopulateApproverInfo
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic
N/A

Modified Logic (Changes are in bold)
<pre> {----- Added routine to allow population of these fields from TIUSign CodeCR563 JRL 6/13/14 -----} procedure TXmlEngine.PopulateApproverInfo(AppID, AppAssnAuth, AppLast, AppFirst, AppMid : String); begin Approver.ApproverID := AppID; Approver.ApproverAssigningAuthority := AppAssnAuth; Approver.ApproverLastName := AppLast; Approver.ApproverFirstName := AppFirst; Approver.ApproverMiddleName := AppMid; end;</pre>

- 6.2.2.7.67. Dialog – Not applicable for CodeCR 563**
- 6.2.2.7.68. Help Frame – Not applicable for CodeCR 563**
- 6.2.2.7.69. HL7 Application Parameter – Not applicable for CodeCR 563**
- 6.2.2.7.70. HL7 Logical Link – Not applicable for CodeCR 563**
- 6.2.2.7.71. COTS Interface – Not applicable for CodeCR 563**
- 6.2.2.8. CODECR568 – Notify VLER of New Exam Request**

- 6.2.2.8.1. Routines (Entry Points) – Not applicable for CodeCR568**
- 6.2.2.8.2. Templates – Not applicable for CodeCR568**
- 6.2.2.8.3. Bulletins – Not applicable for CodeCR568**
- 6.2.2.8.4. Data Entries Affected by the Design – Not applicable for CodeCR568**
- 6.2.2.8.5. Unique Record(s) – Not applicable for CodeCR568**
- 6.2.2.8.6. File or Global Size Changes – Not applicable for CodeCR568**
- 6.2.2.8.7. Mail Groups – Not applicable for CodeCR568**
- 6.2.2.8.8. Security Keys – Not applicable for CodeCR568**
- 6.2.2.8.9. Options – Not applicable for CodeCR568**
- 6.2.2.8.10. Protocols – Not applicable for CodeCR568**
- 6.2.2.8.11. Remote Procedure Call (RPC) – Not applicable for CodeCR568**
- 6.2.2.8.12. Constants Defined in Interface – Not applicable for CodeCR568**
- 6.2.2.8.13. Variables Defined in Interface – Not applicable for CodeCR568**
- 6.2.2.8.14. Types Defined in Interface – Not applicable for CodeCR568**
- 6.2.2.8.15. GUI**

Table 36: GUI

Unit Name	Description
demTranMain.pas	<p>All of the units listed here were modified and some of these listed are newly added.</p> <p>These units combined implement the functionality required by CodeCR568. Many of these units also share common code that is used for CodeCR570.</p> <p>Further Indy Library version 10 was modified to update and add missing function calls needed specifically for VLER/DAS connectivity. Modified files are IdHTTP.pas, IdURI.pas.</p> <p>Additionally encountered bugs and inefficient code in the Indy Library. These were fixed in the library code as well.</p>
CAPRISupport.pas	
clsSFTPConn.pas	
clsVendConn.pas	
clsVLERConn.pas	
frmContractedExamNewResend.pas	
Patient.pas	
UnitDocumentMethods.pas	
untConstVals.pas	
untMiscMthds.pas	
vlerCCRExams.pas	
VlerDasClient.pas	
vlerDasIdHTTP.pas	
XMLTemplate_CapriExamRequest.pas	

6.2.2.8.16. GUI Classes – Not applicable for CodeCR568

6.2.2.8.17. Current Form

The screenshot shows the CAPRI Contract Referral (CCR) application window. The title bar reads "CAPRI Contract Referral (CCR)" with standard window controls. The menu bar includes "File", "Tools", "Reports", "Admin Tools", and "Help".

The main interface is divided into sections:

- Patient Section:** Displays "PatientLast, PatientFirst MI." and "000-00-0000 September 21, 1940 (70)". A "Select Patient" button is located to the right.
- Contracted 2507 Requests and Exams Section:** A large central area containing several icons representing different functions:
 - mnuMain (Menu icon)
 - CCOWRPCBrkrMain (Icon with a blue 'C' and a person)
 - tmrShowForm (Clock icon)
 - tmrEnsrBrkrCnntctd (Clock icon)
 - tmrAppShutDown (Clock icon)
 - fmLstrReq (Icon with 'FM' and a document)
 - fmLstrExam (Icon with 'FM' and a document)
 - fmLstrCntrcts (Icon with 'FM' and a document)
- Referral Status:** A dropdown menu.
- Date Range:** Fields for "From:" and "To:" with calendar icons, and a "Refresh" button.
- Action Buttons:** "Refer Exams", "Resend", and "Download Exam Results" at the bottom.

6.2.2.8.18. Modified Form

The screenshot shows the CAPRI Contract Referral (CCR) application window. The title bar reads "CAPRI Contract Referral (CCR)". The menu bar includes "File", "Tools", "Reports", "Admin Tools", and "Help".

Patient Information:

- Name: PatientLast, PatientFirst MI.
- SSN: 000-00-0000P
- DOB: September 21, 1940 (70)

Buttons: "SimulateVLEResults" and "Select Patient".

Contracted 2507 Requests and Exams

This section contains a diagram of components:

- Left Panel:** A list of components including FMEExamNu, FMFeeExa, FMEExamTy, FMEExamR, FMEExamSt, FMEExamContr, and FMEExamContr.
- Center Area:** A collection of components including FMFile1, fmGetsTemp, mnuMain, CCOWRPCBrkrMain, fmFindOneTemp, tmrShowForm, tmrEnsrBrkrCnntctd, tmrAppShutDwn, fmLstrReq, fmLstrExam, and fmLstrCntrcts.

Referral Status: A dropdown menu.

Date Range: Fields for "From:" and "To:" with calendar icons, and a "Refresh" button.

Buttons: "Refer Exams", "Regend", and "Download Exam Results".

6.2.2.8.19. Components on Form

Table 39: Components on Form

Name	Type	Description
GenMedNewExamPanel	TPanel	A collection of all components needed specifically for GenMed handling
FMEExamNumber	TFMEdit	Used in creating a new Exam

Name	Type	Description
FMFeeExam	TFMEdit	Used in creating a new Exam
FMEExamType	TFMEdit	Used in creating a new Exam
FMEExamRequest	TFMEdit	Used in creating a new Exam
FMEExamStatus	TFMEdit	Used in creating a new Exam
FMEExamContractor	TFMEdit	Used in creating a new Exam
FMEExamDtSent	TFMEdit	Used in creating a new Exam
FMGetsTemp	TFMGets	Used in creating a new Exam
FMFindOneTemp	TFMFindOne	Used in creating a new Exam
FMFiler1	TFMFiler	Used in creating a new Exam

6.2.2.8.20. Events

Table 40: Events

Name	Type	Description
demTranMain.btnNewExamClick	OnClick	Event to trigger the creation of the form used to send new Exams to vendor
demTranMain.btnCheckInExamClick	OnClick	Event to trigger the fetch from VLER, of any pending for review exam results to any requests for the current station from any of the contracted vendors

6.2.2.8.21. Methods

Table 41: Methods

Method Name	Procedure/Function	Description
TExntdCtrctdBaseFormNewRsnd CreateAndSendContractorMsg	Procedure	Dual purpose method to create a message for a contractor/vendor depending upon the vendor configuration
GenerateVLERFileName	Function	Create a new filename to be used to send XML to VLER/DAS
GenerateSftpFileName	Function	Create a new filename to be used to send XML via SFTP to a vendor

Method Name	Procedure/Function	Description
CreateVLERXML	Function	Create the XML for a new exam request to be sent via VLER/DAS
CreateSftpXML	Function	Create the XML for a new exam request to be sent via SFTP
TransmitXMLMessage	Procedure	Dual purpose method to transmit the created XML to either an SFTP destination or VLER/DAS
TfrmCntrctdExmMain ProcessFilesVLER	Function	Processes all files received from VLER and matches them up with open exam requests
ProcessFilesSFTP	Function	Processes all files received via SFTP and matches them up with open exam requests
UpdateVistaVLER	Function	Updates Vista for all files received from VLER that had already been processed.
UpdateVistaSFTP	Function	Updates Vista for all files received via SFTP that had already been processed.
FillContractorsList	Procedure	Fills a temporary in-memory cache of contractors for speed.
SetContractorVLEREnabledAndName	Procedure	Updates a single contractor's values in the above mentioned temporary in-memory cache of contractors for speed.
AttemptToSetInProgress... ManifestsOpenTStringList	Procedure	Resets any InProcess manifests back to Open, that were not used by the current download.
IsContractorVLEREnabled	Function	Gets the contractor's enabled status in the above mentioned temporary in-memory cache of contractors for speed.

Method Name	Procedure/Function	Description
GetContractorName	Function	Gets the contractor's Name as listed in the above mentioned temporary in-memory cache of contractors for speed.
CreateNewExamSentByVendor	Function	Creates a new exam when a vendor returned an exam result but there was no corresponding exam request. Typically it is part of the GenMed type scenario
TfrmVlerCCRExams EnsureTfrmVlerCCRExamsReady	Procedure	Class method that can be called from anywhere to make sure this "utility" form is available for use.
CcrGetManifestCount	Function	Gets count of all open manifests available for current station from VLER/DAS
CcrGetManifests	Procedure	Fetches all open manifests and exams using CcrGetManifestURLsExams
CcrGetManifestURLsExams	Procedure	Fetches exams for a list of open manifests
FetchAllExamsTextToFiles	Procedure	Fetches exam text files uploaded by vendors for a list of open manifests
FetchAFileObjectFromFS	Procedure	Utility method to fetch a GridFS object from VLER/DAS to file
FetchAFileObjectFromFSToStream	Procedure	Utility method to fetch a GridFS object from VLER/DAS to a TStream object
FetchAFileObjectFromVLERToString	Function	Utility method to fetch a GridFS object from VLER/DAS to a String Variable
getStatusDateText	Function	Given any XMLNode, this method finds the status date node at any level and returns the date text

Method Name	Procedure/Function	Description
SetResultStatusText	Procedure	Utility method to Set Result Status text in an XML Document to be sent to vendor via VLER/DAS
SetResultStatusDate	Procedure	Utility method to Set Result Status Date text in an XML Document to be sent to vendor via VLER/DAS
SetManifestStatus	Function	Using a ManifestID, directly update the status node of a manifest in VLER/DAS
GetExamResultByID2DOM	Procedure	Using a GridFS Object id of a result XML Document, fetch it from VLER/DAS and store into an XMLDom object
GetQuerySetOfAllNonTextAttachments	Procedure	Filter all non text attachments from a TXMLDocument object
Unit untMiscMthds.pas		
ValidateTypeAndInstance	Function	Utility method to validate a particular object still exists and its type to help avoid potential run time errors with dangling object references.
FreeAndClearTStrings	Procedure	Utility method to generically clear a Tstring class and free all its associated objects
EmptyListAndFreeObjects	Procedure	Utility method to generically ensure that any TList reference gets all of it's contained nodes and the associated objects properly freed by their corresponding class's free methods. After that it is cleared and freed.
FreeAndNilListWithObjects	Procedure	Calls EmptyListAndFreeObjects and also "nil" the variable passed in.

Method Name	Procedure/Function	Description
EmptyTStringsAndFreeObjects	Procedure	Utility method to generically ensure that any TString reference gets all of it's contained nodes and the associated objects properly freed by their corresponding class's free methods. After that it is cleared and freed.
FreeAndNilTStringsWithObjects	Procedure	Calls EmptyTStringsAndFreeObjects and also "nil" the variable passed in.
SaveStrXMLInFileAndXMLDoc	Procedure	Utility method to help with debugging at run time by writing a String containing XML, into temporary files in a specified temporary folder and also moving it into an TXMLDocument variable. Method also allows runtime override of XML string value.
GenericGetXMLNodeText	Function	Generically get a Node's text value, when exact type of node (or if it has child objects/attributes etc) is not known. Specifically useful for DateTime type fields.
FindNodeByNodeName	Function	Finds a specifically named child node given any root node.
FindNodeInPathByNodeName	Function	Finds a specifically named child node given any root node and path of child nodes to reach final destination node.
FindOrInsertNodeInPathByNodeName	Function	Finds a specifically named child node given any root node and path of child nodes to reach final destination node. If the destination child node is not found then insert it with the given path

Method Name	Procedure/Function	Description
AddChildWithTextValue	Procedure	Add a child node with the given value
UpdateChildWithTextValue	Procedure	Update a child node with the given value and if it doesn't exist Add it.
AddDocumentsWrapperIfMissing	Procedure	Some VLER/DAS queries return result set without the outer wrapper node called <documents>. This method will fix the XML by adding the container if needed.
MakeOutputFileStreamReady	Function	Retruns a TFileStream ready to write to for any given FileName
Unit VlerDasIdHTTP.pas	Manages a TIdHTTP Instance	Provides support for working with proxy server, security token handling, SSL protocol handler, host URL mapping to VLER and support for encoding parameters
URLParamsEncode	Function	Give a full URL string encodes only the parameters
ChangeVlerHostInURL	Function	Optimized function to generically change hostname and port in a URL to the current proxy host and port.
InitVLERBaseRequestHeader	Procedure	Method to initialize the TIdHTTPvler object and security token as well as set other defaults.
Unit clsVendConn.pas		Declares TVendorConnect, a class used as base class for both SFTP and VLER type connections. Provides low level common services
GetVndrConnInfo	Procedure	Get vendor connection values from Vista

Method Name	Procedure/Function	Description
Unit clsSFTPConn.pas		Declares TSFTPConn for SFTP connection specific functions
only refactored method out of this unit into clsVendConn.pas and modified the whole unit so that its class inherits from TVendorConnect		
Unit clsVLERConn.pas		Declares TVLERConn for VLER connection specific functions
GetExamFileListForPatient	Function	Given a patient SSN, this method will get all the exam results available for that patient, fetch the text files for those results and return the exam results in a TList
UploadFileMultiPart	Procedure	Uploads any Exam Rejection and Completion file to VLER/DAS
UploadFile	Procedure	Upload Exam Request to Appropriate Vendor via VLER DAS
StripMSAndZFromISO8601	Procedure	Removes milliseconds and trailing Z from any ISO8601 datetime value

- 6.2.2.8.22. Special References – Not applicable for CodeCR568**
- 6.2.2.8.23. Class Events – Not applicable for CodeCR568**
- 6.2.2.8.24. Class Methods – Not applicable for CodeCR568**
- 6.2.2.8.25. Class Properties – Not applicable for CodeCR568**
- 6.2.2.8.26. Uses Clause – Not applicable for CodeCR568**
- 6.2.2.8.27. Forms – Not applicable for CodeCR568**
- 6.2.2.8.28. Functions – Not applicable for CodeCR568**
- 6.2.2.8.29. Dialog – Not applicable for CodeCR568**
- 6.2.2.8.30. Help Frame – Not applicable for CodeCR568**
- 6.2.2.8.31. HL7 Application Parameter – Not applicable for CodeCR568**
- 6.2.2.8.32. COTS Interface – Not applicable for CodeCR568**

6.2.2.9. CODECR569 – Event Notification “Pending-UnderReview”

Note: See CODECR571 – Event Notification “DBQ Exam Result Manifest Available” which includes Event Notification “Pending-UnderReview”

6.2.2.10. CODECR570 – Notify VLER of Completed or Rejected Exams

Note: See CodeCR568 – Event Notification “Notify VLER of new Exam Request” which also includes all the common information for this CodeCR570 “Notify VLER of Completed or Rejected Exams”.

- 6.2.2.10.1. Routines (Entry Points) – Not applicable for CodeCR570**
- 6.2.2.10.2. Templates – Not applicable for CodeCR570**
- 6.2.2.10.3. Bulletins – Not applicable for CodeCR570**
- 6.2.2.10.4. Data Entries Affected by the Design – Not applicable for CodeCR570**
- 6.2.2.10.5. Unique Record(s) – Not applicable for CodeCR570**
- 6.2.2.10.6. File or Global Size Changes – Not applicable for CodeCR570**
- 6.2.2.10.7. Mail Groups – Not applicable for CodeCR570**
- 6.2.2.10.8. Security Keys – Not applicable for CodeCR570**
- 6.2.2.10.9. Options – Not applicable for CodeCR570**
- 6.2.2.10.10. Protocols – Not applicable for CodeCR570**
- 6.2.2.10.11. Remote Procedure Call (RPC) – Not applicable for CodeCR570**
- 6.2.2.10.12. Constants Defined in Interface – Not applicable for CodeCR570**
- 6.2.2.10.13. Variables Defined in Interface – Not applicable for CodeCR570**
- 6.2.2.10.14. Types Defined in Interface – Not applicable for CodeCR570**
- 6.2.2.10.15. GUI**

Table 36: GUI

Unit Name	Description
managereportsCCR.pas	

6.2.2.10.17. Current Form

CCR - C&P Exam Request Report Management

C&P Exam Requests:

FM FM FM FM FM

FMLister FMGetsEx FMValid FMtmr Vmr Ensr Brkr Cnnctd

Refresh List

Re-open this request

Release this OPEN Request

Exams:

Edit Selected Exam

FMEditRequestStatus

STATUS:

Report text:

mnuPopEdit

EXAM LOCATION:

DATE OF EXAM:

EXAMINING PROVIDER:

Reassign to Vendor


Save Changes

Cancel Changes








Mark Insufficient













Close Window

6.2.2.10.18. Modified Form


CCR - C&P Exam Request Report Management

C&P Exam Requests:

MGFMFindOneTIUTitle

Refresh List

Re-open this request

Release this OPEN Request

Exams:

Edit Selected Exam-not Enabled in CCR
View Exam Attachments



Test


STATUS:

▼

Signature:

Report text:

B

U

A⁺
A⁻


mnuPopEdit

EXAM LOCATION:

▼

DATE OF EXAM:

EXAMINING PROVIDER:

Reassign to Vendor

Save Changes

Cancel Changes

Mark Insufficient

Close Window

6.2.2.10.19. Components on Form

Table 39: Components on Form

Name	Type	Description
ButtonViewExamAttachments	TButton	Fetches the attachments for a particular exam from VLER and shows them to user.

6.2.2.10.20. Events

Table 40: Events

Name	Type	Description
------	------	-------------

Name	Type	Description
ButtonViewExamAttachmentsClick	OnClick	Fetches the attachments for a particular exam from VLER and shows them to user.

6.2.2.10.21. Methods

Table 41: Methods

Method Name	Procedure/Function	Description
BuildRejectionXMLForVLERFromSFTPStyleXML	Function	
TransmitXMLMessageRejectSFTP	Procedure	
TransmitXMLMessageRejectVLER	Procedure	
PrepExamResultForFindCopyInsert	Procedure	
UploadExamUpdateToVLERCon	Procedure	
TransmitXMLMessageCompleteVLER	Procedure	
LoadExams	Function	
RemoveResultIDFromExam	Procedure	
AddChildWithTextValue	Procedure	
ButtonReleaseClick	Procedure	Added call to ReleaseVendorExamResultsToVler to send vendor exam results to VLER DAS.
ReleaseVendorExamResultsToVLER	Function	function TfrmManageReportsCCR.ReleaseVendorExamResultsToVLER(requestID: string): boolean; For exam request requestID, send completed vendor exam results to VLER DAS. VBMS will pick up the results from VLER DAS. This function is extracted from a section of code that was originally in ButtonReleaseClick.

6.2.2.10.22. GUI

Table 36: GUI

Unit Name	Description
Tiusign.pas	Sign in house exam results

6.2.2.10.23. GUI Classes – Not applicable for CodeCR570

6.2.2.10.24. Current Form

N/A

6.2.2.10.25. Modified Form

N/A no change

6.2.2.10.26. Components on Form

Table 39: Components on Form

Name	Type	Description

6.2.2.10.27. Events

Table 40: Events

Name	Type	Description

6.2.2.10.28. Methods

Table 41: Methods

Method Name	Procedure/F unction	Description
ButtonOK2Click	Procedure	Added call to ReleaseVendorExamResultsToVLER if IsExamOKtoRelease.
ReleaseVendorExamR esultsToVLER	Function	function TTIUSignForm.ReleaseVendorExamResultsToVLER(selectedRequestID: string): Boolean; For exam request selectedRequestID, instantiate frmCntrctdExmMain, then instantiate frmManageReportsCCR, then call frmManageReportsCCR.ReleaseVendorExamResults ToVLER(selectedRequestID); to send completed vendor exam results to VLER DAS. VBMS will pick up the results from VLER DAS.

- 6.2.2.10.29. Special References – Not applicable for CodeCR570**
- 6.2.2.10.30. Class Events – Not applicable for CodeCR570**
- 6.2.2.10.31. Class Methods – Not applicable for CodeCR570**
- 6.2.2.10.32. Class Properties – Not applicable for CodeCR570**
- 6.2.2.10.33. Uses Clause – Not applicable for CodeCR570**
- 6.2.2.10.34. Forms – Not applicable for CodeCR570**
- 6.2.2.10.35. Functions – Not applicable for CodeCR570**
- 6.2.2.10.36. Dialog – Not applicable for CodeCR570**
- 6.2.2.10.37. Help Frame – Not applicable for CodeCR570**
- 6.2.2.10.38. HL7 Application Parameter – Not applicable for CodeCR570**
- 6.2.2.10.39. COTS Interface – Not applicable for CodeCR570**

6.2.2.11. CODECR571 – Event Notification "DBQ Exam Result Manifest Available"

- 6.2.2.11.1. Routines (Entry Points) – Not Applicable for CodeCR 571**
- 6.2.2.11.2. Templates – Not Applicable for CodeCR 571**
- 6.2.2.11.3. Bulletins – Not Applicable for CodeCR 571**
- 6.2.2.11.4. Data Entries Affected by the Design – Not Applicable for CodeCR 571**
- 6.2.2.11.5. Unique Record(s) – Not Applicable for CodeCR 571**
- 6.2.2.11.6. File or Global Size Changes – Not Applicable for CodeCR 571**
- 6.2.2.11.7. Mail Groups – Not Applicable for CodeCR 571**
- 6.2.2.11.8. Security Keys – Not Applicable for CodeCR 571**
- 6.2.2.11.9. Options – Not Applicable for CodeCR 571**
- 6.2.2.11.10. Protocols – Not Applicable for CodeCR 571**
- 6.2.2.11.11. Remote Procedure Call (RPC) – Not Applicable for CodeCR 571**
- 6.2.2.11.12. Constants Defined in Interface – Not Applicable for CodeCR 571**
- 6.2.2.11.13. Variables Defined in Interface – Not Applicable for CodeCR 571**
- 6.2.2.11.14. Types Defined in Interface – Not Applicable for CodeCR 571**

6.2.2.11.15. CODECR571 - GUI

6.2.2.11.16. GUI - Alerts

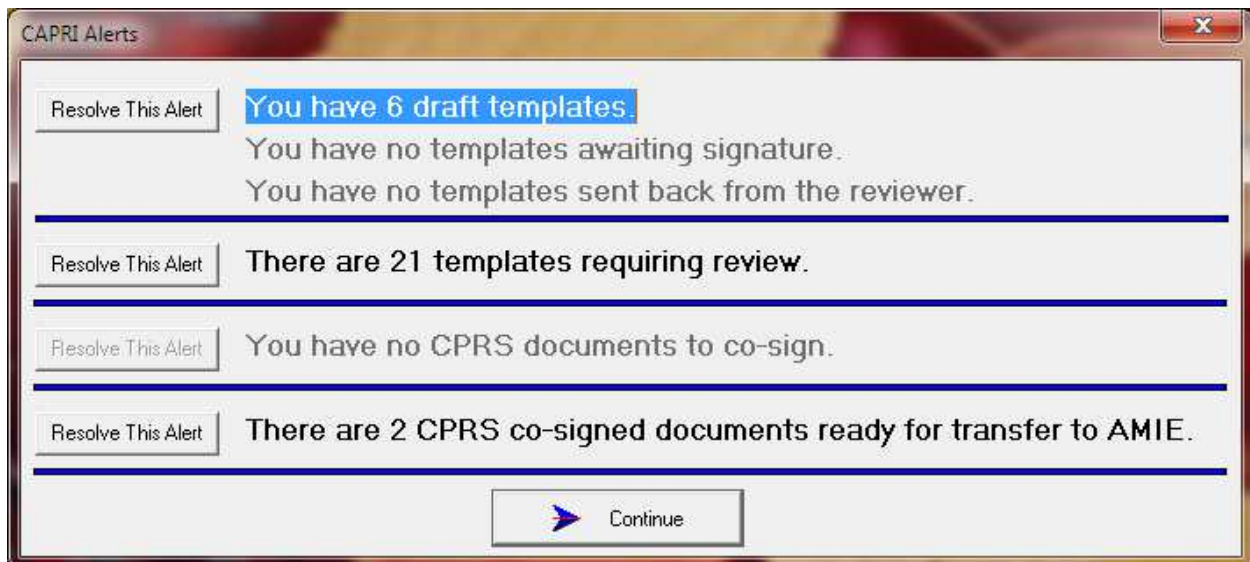
Unit Name	Description
Alerts.pas	A list of actions waiting for user input.

6.2.2.11.17. GUI Classes

Table 38: GUI Classes

GUI Classes	Alerts
Class Name	TfrmAlerts
Derived From Class	TForm
Purpose	Display notifications of pending actions

6.2.2.11.18. Current Form



The screenshot shows a Windows-style dialog box titled "CAPRI Alerts". It contains four alert items, each with a "Resolve This Alert" button on the left and a text message on the right. The first alert is highlighted with a blue background and reads "You have 6 draft templates." followed by "You have no templates awaiting signature." and "You have no templates sent back from the reviewer." The second alert reads "There are 21 templates requiring review." The third alert reads "You have no CPRS documents to co-sign." The fourth alert reads "There are 2 CPRS co-signed documents ready for transfer to AMIE." At the bottom of the dialog is a "Continue" button with a blue arrow icon.

6.2.2.11.19. Modified Form

The image shows a 'CAPRI Alerts' dialog box with a title bar and a close button. It contains five alert items, each with a 'Resolve This Alert' button and a message. The messages are: 'You have no draft templates.', 'You have no templates awaiting signature.', 'You have no templates sent back from the reviewer.', 'There are no templates requiring review.', and 'You have 1 CPRS document to co-sign.' (highlighted in blue). The last message is followed by 'There are no CPRS co-signed documents ready for transfer to AMIE.' and 'You have 4 vendor exam requests pending review.'. At the bottom is a 'Continue' button with a blue arrow icon.

6.2.2.11.20. Components on Form

Table 39: Components on Form

Name	Type	Description
btnManifests	TButton	Click this button to display vendor results pending review.

6.2.2.11.21. Events

Table 40: Events

Name	Type	Description
btnManifests	OnClick	Display vendor results pending review.

6.2.2.11.22. Methods

Table 41: Methods

Method Name	Procedure/Function	Description
btnmanifestsClick	Procedure	Show Manifests form modally.

6.2.2.11.23. Special References

Special Reference Name	Type	Description

6.2.2.11.24. Class Events

Table 42: Class Events

Name	Type	Description

6.2.2.11.25. Class Methods

Table 43: Class Methods

Name	Procedure/Function	Description

6.2.2.11.26. Class Properties

Table 44: Class Properties

Class Properties Name	Type	Visibility	Description

6.2.2.11.27. Uses Clause

6.2.2.11.28. GUI - Main

Unit Name	Description
Main	Main program unit

6.2.2.11.29. GUI Classes

Table 38: GUI Classes

GUI Classes	Instructions
Class Name	TfrmMain
Derived From Class	TForm
Purpose	Main program

6.2.2.11.30. Current Form

N/A

6.2.2.11.31. Modified Form

N/A no graphical changes.

6.2.2.11.32. Components on Form

Table 39: Components on Form

Name	Type	Description

6.2.2.11.33. Events

Table 40: Events

Name	Type	Description

6.2.2.11.34. Methods

Table 41: Methods

Method Name	Procedure/Function	Description

6.2.2.11.35. Special References

Special Reference Name	Type	Description
TPatientInfoBucket	Class	Moved from main to Patients unit

6.2.2.11.36. Class Events

Table 42: Class Events

Name	Type	Description
------	------	-------------

Name	Type	Description

6.2.2.11.37. Class Methods

Table 43: Class Methods

Name	Procedure/Function	Description
actFileConnectExecute	Procedure	Added code to get manifest count and display a corresponding message in the alerts form: "You have n vendor results pending review."
GetCurrentPatientInfo	Function	Modified to call TPatientInfoBucket.Create(PatientSSN) if FPatientInfoBucket is nil.

6.2.2.11.38. Class Properties

Table 44: Class Properties

Class Properties Name	Type	Visibility	Description

6.2.2.11.39. Uses Clause

6.2.2.11.40. GUI - Manifests

Unit Name	Description
Manifests.pas	Gathers manifests and vendor exam results that are pending review for a given station. Displays a list of patients and exams. User is able to select a patient and request the CAPRI Contract Referral (CCR) form to process the exam results pending review for that patient.

6.2.2.11.41. GUI Classes

Table 38: GUI Classes

GUI Classes	Manifests
Class Name	TfrmManifests
Derived From Class	TForm
Purpose	Gathers manifests and vendor exam results that are pending review for a given station. Displays a list of patients and exams. User is able to select a patient and request the CAPRI Contract Referral (CCR) form to process the exam results pending review for that patient.

6.2.2.11.42. Current Form

N/A new form

6.2.2.11.43. Modified Form

Manifests Pending Under Review

Patient	Exam Name
ROBERT, JOHN	DBQ AUDIO Hearing loss & tinnitus
	DBQ ENDO Diabetes Mellitus
	DBQ CARDIO Ischemic Heart Disease Exam Report V4
	DBQ DERM Scars Exam Report V4
WATER, SAM	DBQ MUSC Neck (cervical spine)
	DBQ MUSC Ankle
	DBQ MUSC Knee and Lower Leg
	DBQ MUSC Shoulder & Arm
	DBQ NEURO HEADACHES (INCLUDING MIGRAINE HEADACHES)
	DBQ MUSC Ankle
	DBQ MUSC Knee and Lower Leg
TAYLOR, LILLY	DBQ AUDIO Hearing loss & tinnitus
	DBQ CARDIO Heart
	DBQ CARDIO Hypertension
	DBQ DERM Scars

Pending Exam Results

Capri Contract Referral (CCR)

Close

Patient Name: WATER, SAM

6.2.2.11.44. Components on Form

Table 39: Components on Form

Name	Type	Description
btnPendExamResults	TButton	Click this button to display vendor results pending review.
btnGetPatient	TButton	Retrieve info for selected patient. If Patient found in Vista, enable btnCCR. Button is hidden. Used in code only.
btnCCR	TButton	Launch CAPRI Contract Referral (CCR) form for selected patient.
lbxIPR	TListBox	Displays patients and associated exams.

6.2.2.11.45. Events

Table 40: Events

Name	Type	Description
btnPendExamResults	OnClick	Display vendor results pending review.
btnGetPatient	OnClick	Get patient info for selected patient.
btnCCR	OnClick	For selected patient, launch CAPRI Contract Referral (CCR) form.
lbxIPR	OnClick	Row is clicked

6.2.2.11.46. Methods

Table 41: Methods

Method Name	Procedure/Function	Description
btnPendExamResultsClick	Procedure	Refresh list of manifests and exam results pending review. Display patients and associated exams.
btnGetPatientClick	Procedure	For selected patient SSN, populate a patientinfobucket object. If patient found, enable btnCCR.

Method Name	Procedure/Function	Description
btnCCRClick	Procedure	Call btnGetPatient.Click to populate patient. Call main form CCRLaunchExecute to display CAPRI Contract Referral (CCR) form.
lbxIPRClick	Procedure	If selected row contains an exam, scroll up to its patient row. Then, call btnGetPatient.Click

6.2.2.11.47. Special References

Special Reference Name	Type	Description

6.2.2.11.48. Class Events

Table 42: Class Events

Name	Type	Description

6.2.2.11.49. Class Methods

Table 43: Class Methods

Name	Procedure/Function	Description
TfrmManifests.getManifestList	Procedure	For user's station, query VLER DAS manifest notifications available and store in XMLManifestDoc. If PARAM_SAVE_XML true, remove old temp xml files, save Manifests.xml to the user's temporary CAPRI folder.
TfrmManifests.getManifestDoc	Procedure	For manifest URI, retrieve manifest doc from VLER DAS. If PARAM_SAVE_XML true, save ManifestDoc_n.xml to the user's temporary CAPRI folder.
TfrmManifests.getPatientExams	Procedure	For XMLManifestDoc, retrieve patient info and exam info and store the data in listbox lbxIPR.
SSNTolssn	Function	Converts string SSN to integer SSN. Pseudo SSN with trailing P has 1 billion added to its integer value.

Name	Procedure/Function	Description
TfrmManifests.getManifestURL	Procedure	If XMLManifests not active, call getManifestList. In XMLManifest, parse ManifestURL. Call getManifestDoc with ManifestURL. Call getPatientExams.
TfrmManifests.getManifestCnt	Function	Call getManifestList. Return number of top-level nodes in XMLManifests. Called by main to update alerts form.
ISSNToSSN	Function	Convert integer SSN to string SSN. If input is greater than 1 billion, reduce value by that before converting to string and append P for pseudo SSN.
RemoveFile	Procedure	For inManifestFile input, deletes the file in the user's temporary CAPRI directory.
RemoveManifestFiles	Procedure	Finds and deletes the manifests.xml and the set of manifestdoc_n.xml files generated by a previous run of getManifestList.

6.2.2.11.50. Class Properties

Table 44: Class Properties

Class Properties Name	Type	Visibility	Description

6.2.2.11.51. Uses Clause

Interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Fmctrlns, xmldom, XMLIntf, msxmldom, XMLDoc, ComCtrls, IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient, IdHTTP, Fmcmpnts, Diaccess, VlerDasIdHTTP, VlerDasClient;

Implementation

uses

main, clsPatientInfo, Patient , IdURI , untBrkrMthds, untMiscMthds, MFunStr;

6.2.2.11.52. GUI - Patients

Unit Name	Description
Patients.pas	Unit which retrieves information from the Patient file for a SSN and allocates a TPatientInfoBucket object where the data is stored.

6.2.2.11.53. GUI Classes

Table 38: GUI Classes

GUI Classes	PatientInfoBucket
Class Name	TPatientInfoBucket
Derived From Class	TObject
Purpose	Retrieves information from the Patient file for a SSN and allocates a TPatientInfoBucket object where the data is stored.

6.2.2.11.54. Current Form

N/A formerly a unit, no form used.

6.2.2.11.55. Modified Form

N/A - no visual components.

6.2.2.11.56. Components on Form

Table 39: Components on Form

Name	Type	Description
FMGets1	TFMGets	Gets a record in the Patient file and selected field numbers for a given IEN.
FMFindOne1	TFMFindOne	Searches for a Patient record for a given SSN and returns the IEN if found.

6.2.2.11.57. Events

Table 40: Events

Name	Type	Description

6.2.2.11.58. Methods

Table 41: Methods

Method Name	Procedure/Function	Description
-------------	--------------------	-------------

Method Name	Procedure/Function	Description

6.2.2.11.59. Special References

Special Reference Name	Type	Description

6.2.2.11.60. Class Events

Table 42: Class Events

Name	Type	Description

6.2.2.11.61. Class Methods

Table 43: Class Methods

Name	Procedure/Function	Description
TPatientInfoBucket.Create(inSSN: String)	Constructor	With SSN input, searches for Patient record and if found, stores Patient info in allocated TPatientInfoBucket structure. Creates frmPatient in order to use FileMan components FMGets1 and FMFindOne1.
monthNum	Function	For 3 character month abbreviation, returns ordinal number of month.
encTextToDate	Function	Convert string date input to TDateTime data.

6.2.2.11.62. Class Properties

Table 44: Class Properties

Class Properties Name	Type	Visibility	Description

6.2.2.11.63. Uses Clause

Interface Uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, FMCompnts, Diaccess

Implementation Uses

Mfunstr

6.2.2.12. CODECR577 – VVA Security Keys

6.2.2.12.1. Routines (Entry Points)

Table 15 (Grouping): Routines

Routines	Activities
Routine Name	DVBA187P
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
RTM	2.6.11
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	N/A

Routines	Activities
Data Dictionary (DD) References	#200 NEW PERSON #19.1 SECURITY KEYS
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	#200
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output Reference <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input checked="" type="checkbox"/> Local
Input Attribute Name and Definition	Name: Definition:
Output Attribute Name and Definition	Name: Definition:

Current Logic
N/A

Modified Logic (Changes are in bold)

Modified Logic (Changes are in bold)

DVBA187P ;ALB/GAK - PATCH DVBA*2.7*187 POST-INSTALL ROUTINE;08-OCT-2013
;;2.7;AMIE;**187**;Apr 10, 1995

; This routine adds an entry to the REMOTE APPLICATION file (#8994.5) for VLER
DAS-CAPRI

Q

;

ENTER ;

D AMIE

D SECKEY

Q

;

;

AMIE ;Update for the AMIE EXAM (#396.6) file

;

;Used to inactivate old entries and/or create new entries for

;designated worksheet updates

;

D BMES^XPDUTL(" Update to AMIE EXAM (#396.6) file...")

I '\$D(^DVB(396.6)) D BMES^XPDUTL("Missing AMIE EXAM (#396.6) file") Q

I '\$D(^DVB(396.6)) D

. D NEW

Q

;

;

NEW ;Add new exam entry

;

N DVBAI,DVBLINE,DVBIEN,DVBEXM,DVBPNM,DVBBDY,DVBROU,DVBSTAT,DVBWKS

;

D BMES^XPDUTL(" Adding new AMIE EXAM (#396.6) file entry...")

F DVBAI=1:1 S DVBLINE=\$P(\$T(AMIENew+DVBAI),",",2) Q:DVBLINE="QUIT" D

. N DVBA MSG

. S DVBIEN=\$P(DVBLINE,",",1) ;ien

. S DVBEXM=\$P(DVBLINE,",",2) ;exam name

. S DVBPNM=\$P(DVBLINE,",",3) ;print name

. S DVBBDY=\$P(DVBLINE,",",4) ;body system

. S DVBROU=\$P(DVBLINE,",",5) ;routine name

. S DVBSTAT=\$P(DVBLINE,",",6) ;status

. S DVBWKS=\$P(DVBLINE,",",8) ;worksheet number

. D BMES^XPDUTL(" Attempting to add Entry #"_DVBIEN_"...")

. D

NEWEXAM^DVBAUTLP(DVBIEN,DVBEXM,DVBPNM,DVBBDY,DVBROU,DVBSTAT,DVBWK

S,DVBAMSG)

CAPRI System Design Document

160

December 2014

; ; Display status message returned if any

. D:\$D(DVBAMSG)>0 MES^XPDUTL(.DVBAMSG)

Q

- 6.2.2.12.2. Templates – Not applicable for CodeCR 577
- 6.2.2.12.3. Bulletins – Not applicable for CodeCR 577
- 6.2.2.12.4. Data Entries Affected by the Design – Not applicable for CodeCR 577
- 6.2.2.12.5. Unique Record(s) - Not applicable for CodeCR 577
- 6.2.2.12.6. File or Global Size Changes – Not applicable for CodeCR 577
- 6.2.2.12.7. Mail Groups – Not applicable for CodeCR 577
- 6.2.2.12.8. Security Keys

Table 26: Security Keys

Security Keys	Activities
Security Key Name	DVBA CAPRI GETVBADOCS
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
SECKEY^DVBA187P	Post install routine	N/A

Security Keys	Activities
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Security Key Description	DVBA CAPRI GETVBADOCS IS A SECURITY KEY FOR CAPRI USERS TO VIEW THE "Get Docs from Virtual VA" OPTION IN CAPRI
Subordinate Keys	N/A
Mutually Exclusive Keys	N/A
Granting Condition Logic	N/A

Current Logic
N/A

Modified Logic (Changes are in bold)
N/A

Security Keys	Activities
---------------	------------

Security Keys	Activities
Hierarchical Precedence	N/A

Security Keys	Activities
Security Key Name	DVBA CAPRI DENY GETVBADOCS
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input checked="" type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
N/A		

Security Keys	Activities
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Security Key Description	This key DENIES users who have this key access to the menu option "Get Docs from VVA".
Subordinate Keys	N/A
Mutually Exclusive Keys	N/A
Granting Condition Logic	N/A

Current Logic
N/A

Security Keys	Activities
Hierarchical Precedence	N/A

- 6.2.2.12.9. Options – Not applicable for CodeCR 577
- 6.2.2.12.10. Protocols – Not applicable for CodeCR 577
- 6.2.2.12.11. Remote Procedure Call (RPC) – Not applicable for CodeCR 577
- 6.2.2.12.12. Constants Defined in Interface – Not applicable for CodeCR 577
- 6.2.2.12.13. Variables Defined in Interface – Not applicable for CodeCR 577
- 6.2.2.12.14. Types Defined in Interface – Not applicable for CodeCR 577
- 6.2.2.12.15. GUI

Table 36: GUI

Unit Name	Description
	Main.pas

- 6.2.2.12.16. GUI Classes – Not applicable for CodeCR 577
- 6.2.2.12.17. Current Form – Not applicable for CodeCR 577
- 6.2.2.12.18. Modified Form – Not applicable for CodeCR 577
- 6.2.2.12.19. Components on Form – Not applicable for CodeCR 577
- 6.2.2.12.20. Events

Table 40: Events

Name	Type	Description
actFileConnectExecute	Procedure	Routine modified to control menu item visibility based on the security key at startup.

- 6.2.2.12.21. Methods – Not applicable for CodeCR 577
- 6.2.2.12.22. Special References – Not applicable for CodeCR 577
- 6.2.2.12.23. Class Events – Not applicable for CodeCR 577
- 6.2.2.12.24. Class Methods – Not applicable for CodeCR 577
- 6.2.2.12.25. Class Properties – Not applicable for CodeCR 577
- 6.2.2.12.26. Uses Clause – Not applicable for CodeCR 577
- 6.2.2.12.27. Forms – Not applicable for CodeCR 577
- 6.2.2.12.28. Functions

Table 48: Functions

Function Name	actFileConnectExecute
Short Description	Modify procedure actFileConnectExecute to update reading of the security keys.

Function Name	actFileConnectExecute
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	N/A

Function Name	actFileConnectExecute
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition:
Output Attribute Name and Definition	Name: N/A Definition:

Current Logic
<pre> if IsUserKeyInList('DVBA CAPRI DENY GETVBADOCS') then // CodeCR497 - JRL 6/07/13 actFileRetrieveVirtualVA.Visible : FALSE // CodeCR497 - JRL 6/07/13 </pre>

Modified Logic (Changes are in bold)

Modified Logic (Changes are in bold)

```
// DVBA CAPRI DENY GETVBADOCs was added in patch 186. VBA users could
// be assigned this key. If this key was assigned, then the menu option
// 'Get Docs from VVA' would not be visible. After implementation, this
// was determined to be hard to administrate, so this key is being
// deprecated and a new key will be added in patch 187 to all users
// to allow them access to 'Get Docs from VVA'. VBA users not allowed
// to see VVA docs here will not get this key. -JRL 5/15/14

if IsPatchInstalled('DVBA*2.7*186') AND // CodeCR577 - JRL 5/15/14 //Only execute this code if
NOT IsPatchInstalled('DVBA*2.7*187') then //Patch 186 is installed and
begin // CodeCR577 - JRL 5/15/14 //Patch 187 is not installed
  if IsUserKeyInList('DVBA CAPRI DENY GETVBADOCs') then // CodeCR497 - JRL 6/07/13
    actFileRetrieveVirtualVA.Visible := FALSE // CodeCR497 - JRL 6/07/13
  else // CodeCR577 - JRL 5/15/14
    actFileRetrieveVirtualVA.Visible := TRUE; // CodeCR577 - JRL 5/15/14
  end; // CodeCR577 - JRL 5/15/14
if IsPatchInstalled('DVBA*2.7*187') then // CodeCR577 - JRL 5/15/14 //Only
execute this code if
begin // CodeCR577 - JRL 5/15/14 //if Patch 187 is installed
  if IsUserKeyInList('DVBA CAPRI GETVBADOCs') then // CodeCR577 - JRL 5/15/14
    actFileRetrieveVirtualVA.Visible := TRUE // CodeCR577 - JRL 5/15/14
  else // CodeCR577 - JRL 5/15/14
    actFileRetrieveVirtualVA.Visible := FALSE; // CodeCR577 - JRL 5/15/14
  end; // CodeCR577 - JRL 5/15/14
end; // CodeCR577 - JRL 5/15/14
```

6.2.2.12.29. Dialog – Not applicable for CodeCR 577

6.2.2.12.30. Help Frame – Not applicable for CodeCR 577

6.2.2.12.31. HL7 Application Parameter – Not applicable for CodeCR 577

6.2.2.12.32. HL7 Logical Link – Not applicable for CodeCR 577

6.2.2.12.33. COTS Interface – Not applicable for CodeCR 577

6.2.2.13. CODECR544 – Add “DBQ General Medical Separation Health Assessment” to the AMIE Exam File

6.2.2.13.1. Routines (Entry Points)

Table 15 (Grouping): Routines

Routines	Activities
Routine Name	DVBA187P
Enhancement Category	<input type="checkbox"/> New <input checked="" type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
RTM	RSD 2.6.12 Add “DBQ Medical SHA” to the AMIE Exam File (Code CR 544)
Related Options	N/A

Related Routines	Routines “Called By”	Routines “Called”
	N/A	N/A

Routines	Activities
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output Reference <input type="checkbox"/> Both <input checked="" type="checkbox"/> Global Reference <input type="checkbox"/> Local
Input Attribute Name and Definition	The post installation routine DVBA187P adds the “DBQ Medical SHA” to the AMIE Exam File
Output Attribute Name and Definition	N/A

Current Logic
N/A

Modified Logic (Changes are in bold)
<pre> DVBA187P ;ALB/GAK - PATCH DVBA*2.7*187 POST-INSTALL ROUTINE;08-OCT-2013 ;;2.7;AMIE;**187**;Apr 10, 1995 Q ; ENTER ; ; D AMIE D SECKEY Q ; ; AMIE ;Update for the AMIE EXAM (#396.6) file ; </pre>

Modified Logic (Changes are in bold)

```

DVBA187P (CONTINUED)
;Used to inactivate old entries and/or create new entries for designate
d worksheet updates
;
D BMES^XPDUTL(" Update to AMIE EXAM (#396.6) file...")
I '$D(^DVB(396.6)) D BMES^XPDUTL("Missing AMIE EXAM (#396.6) file")
Q
I '$D(^DVB(396.6)) D
. ;Add new SHA entry to AMIE EXAM file
. D NEW
. ;Rename existing Medical Opinion 1 entry in AMIE EXAM file
. D RENAMIE
. ;Inactivate existing entires in AMIE EXAM file
. D INACAMIE
. ; Rename Medical Opinion PRINT NAME field
. D RENMEDOP
Q
;
;
NEW ;Add new exam entry
;
N DVBAI,DVBLINE,DVBIEN,DVBEXM,DVBPNM,DVBBDY,DVBROU,DVBSTAT,DVBWKS
;
D BMES^XPDUTL(" Adding new AMIE EXAM (#396.6) file entry...")
F DVBAI=1:1 S DVBLINE=$P($T(AMIENew+DVBAI),";",2) Q:DVBLINE="QUIT"
D
. N DVBAMSG
. S DVBIEN=$P(DVBLINE,";",1) ;ien
. S DVBEXM=$P(DVBLINE,";",2) ;exam name
. S DVBPNM=$P(DVBLINE,";",3) ;print name
. S DVBBDY=$P(DVBLINE,";",4) ;body system
. S DVBRou=$P(DVBLINE,";",5) ;routine name
. S DVBSTAT=$P(DVBLINE,";",6) ;status
. S DVBWKS=$P(DVBLINE,";",8) ;worksheet number
. D BMES^XPDUTL(" Attempting to add Entry #" DVBIEN "...")
. D
NEWEXAM^DVBAUTLP(DVBIEN,DVBEXM,DVBPNM,DVBBDY,DVBROU,DVBSTAT,DVBWKS,
.DVBAMSG)

. ; Display status message returned if any
. D:$D(DVBAMSG)>0 MES^XPDUTL(.DVBAMSG)
. D BMES^XPDUTL(" Completed adding new AMIE EXAM (#396.6) file
entry...")
Q
;
RENAMIE ;Rename existing DBQ exam file entries
;
N DVBAI,DVBLINE,DVBIEN,DVBEXMO,DVBEXMN
;
D BMES^XPDUTL("Renaming AMIE EXAM (#396.6) file entries...")

```

Modified Logic (Changes are in bold)

```

DVBA187P (CONTINUED)
    F DVBAI=1:1 S DVBLINE=$P($T(EXOLDNEW+DVBAI),";";",2) Q:DVBLINE="QUIT"
D
    . S DVBIEN=$P(DVBLINE,";",1) ;ien
    . S DVBEXMO=$P(DVBLINE,";",2) ;old exam name
    . S DVBEXMN=$P(DVBLINE,";",3) ;new exam name
    . D RENEXAM
    D BMES^XPDUTL("Completed Renaming AMIE EXAM (#396.6) file
entries...")
    K DVBEXMO,DVBEXMN
    Q
    ;
RENEXAM ;
    ;Quit if critical variables missing. For each EXOLDNEW entry, do
this.
    I $G(DVBIEN)'>0!($G(DVBEXMO)']"")!($G(DVBEXMN)']"") D Q
    . D BMES^XPDUTL("Insufficient data to process change at
#" DVBIEN ")")
    ;
    ; Update existing entry
    ;
    N DVBAERR,DVBAFDA
    ;
    ; Check for existing entry
    I $G(^DVB(396.6,DVBIEN,0))']" D Q
    . D BMES^XPDUTL("No entry found at #" DVBIEN)
    ;
    ; Check for previous update
    I $P(^DVB(396.6,DVBIEN,0),"^",1)=DVBEXMN D Q
    . D BMES^XPDUTL("Entry at ien #" DVBIEN " has previously been
updated")
    ;
    ; Check for correct entry NAME to update
    I $P(^DVB(396.6,DVBIEN,0),"^",1)'=DVBEXMO D Q
    . D BMES^XPDUTL("Entry at ien #" DVBIEN " does not match expected
name " DVBEXMO " No updating will take place")
    ;
    ; Update entry
    S DVBAFDA(396.6,+DVBIEN ",",.01)=$G(DVBEXMN) D
    . D FILE^DIE("","DVBAFDA","DVBAERR")
    ;
    ; Report sucessful update
    ;
    I $D(DVBAERR("DIERR"))'>0 D Q
    . D BMES^XPDUTL("Renamed entry #" DVBIEN " from " DVBEXMO " to
" DVBEXMN)
    ;
    ; Report update error
    ;
    I $D(DVBAERR("DIERR"))>0 D
    . D BMES^XPDUTL(" *** Warning - Unable to update entry #" DVBIEN "
*** ")

```

Modified Logic (Changes are in bold)

```
DVBA187P (CONTINUED)
    . D MSG^DIALOG()
    Q
    ;
INACAMIE    ;Inactivate exams
    ;
    N DVBAI,DVBLINE,DVBIEN,DVBEXM
    ;
    D BMES^XPDUTL(" Inactivating AMIE EXAM (#396.6) file entries...")
    D MES^XPDUTL("")
    F DVBAI=1:1 S DVBLINE=$P($T(AMIEOLD+DVBAI),";",2) Q:DVBLINE="QUIT"
D
    . N DVBAMSG
    . S DVBIEN=$P(DVBLINE,";",1)
    . S DVBEXM=$P(DVBLINE,";",2)
    . ;D BMES^XPDUTL("Going to INACTEXM^DVBAUTLP with DVBIEN=" DVBIEN ",
DVBEXM=" DVBEXM ", and the message array passed")
    . D INACTEXM^DVBAUTLP(DVBIEN,DVBEXM,.DVBAMSG)
    . ; Display status message returned, if any
    . D:$D(DVBAMSG)>0 MES^XPDUTL(.DVBAMSG)
    . D MES^XPDUTL("")
    D BMES^XPDUTL(" Completed Inactivating AMIE EXAM (#396.6) file
entries...")
    Q
    ;
RENMEDOP    ;
    D BMES^XPDUTL(" Changing PRINT NAME of DBQ Medical Opinion to DBQ
MEDICAL OPINION")
    I $P($G(^DVB(396.6,437,0)), "^",1)'="DBQ Medical Opinion" D Q
    . D BMES^XPDUTL(" Could not change PRINT NAME of DBQ Medical Opinion
to DBQ MEDICAL OPINION")
    N DVBAERR
    S DVBAFDA(396.6,437 " ",",6)="DBQ MEDICAL OPINION" D
FILE^DIE(" ", "DVBAFDA", "DVBAERR")
    I $D(DVBAERR("DIERR"))>0 D
    . D BMES^XPDUTL("DBQ Medical Opinion print name changed to DBQ
MEDICAL OPINION")
    I $D(DVBAERR("DIERR"))>0 D
    . D BMES^XPDUTL("Could not change DBQ Medical Opinion print name to
DBQ MEDICAL OPINION")
    Q
    ;
SECKEY      ;
    ;
    N
XDUZ,KEYNUM,XIEN,XMNU,STOP1,ZTST,OPTIEN,PERDUZ,MSG,ERR,KEYIEN,PERSON,
TODAY,X,ZZ
    ;
    S ZZ="" D OWNSKEY^XUSRB(.ZZ, "XUMGR",DUZ)
    I $G(ZZ(0))'=1 D Q
    . D BMES^XPDUTL("NOTE: THE NEW SECURITY KEY 'DVBA CAPRI GETVBADOCS'
DID NOT SUCCESSFULLY UPDATE WITH THE REQUIRED HOLDERS.")
```

Modified Logic (Changes are in bold)

```
DVBA187P (CONTINUED)
. D BMES^XPDUTL("THE USER RUNNING THIS POST INSTALL ROUTINE DOES NOT
HAVE XUMGR KEY ASSIGNED TO THEM.")
. D BMES^XPDUTL("PLEASE RUN SECKEY^DVBA187P AGAIN WITH USER WHO IS A
HOLDER OF THE 'XUMGR' SECURITY KEY.")
;
K ^TMP($J,"DVBA187P")
;
D NOW^%DTC S TODAY=X
;
;FIND DVBA CAPRI GUI IN OPTION FILE (SHOULD ALWAYS BE 9510) BUT
CHECKING JUST THE SAME
S STOP1=0,OPTIEN=""
S XIEN=0 F S XIEN=$0(^DIC(19,XIEN)) Q:XIEN="!"('XIEN)!(STOP1=1) D
. S ZTST=$G(^DIC(19,XIEN,0),"")
. S ZTST=$P(ZTST,"^",1)
. I ZTST="DVBA CAPRI GUI" S STOP1=1,OPTIEN=XIEN
I OPTIEN="" D BMES^XPDUTL("'DVBA CAPRI GUI' OPTION NOT FOUND IN
OPTION FILE. USERS OF DVBA CAPRI GETVBADOCS COULD NOT BE SETUP") Q
;
;FIND PERSONS WITH DVBA CAPRI GUI OPTION
I OPTIEN'="" D
. S PERDUZ=0 F S PERDUZ=$0(^VA(200,PERDUZ)) Q:PERDUZ="!"('PERDUZ)
D
.. K MSG,ERR
.. D GETS^DIQ(200,PERDUZ "","",9.2,"I","MSG","ERR")
.. I
$G(MSG(200,PERDUZ "","",9.2,"I"))'="",($G(MSG(200,PERDUZ "","",9.2,"I"
))<=TODAY) D Q
... S ^TMP($J,"DVBA187P",PERDUZ,"TERMEDPERSON")=""
.. ;
.. I $G(^VA(200,PERDUZ,201)) I $P(^VA(200,PERDUZ,201),"^",1)=OPTIEN
S ^TMP($J,"DVBA187P",PERDUZ,"USERSWITHOPTION")="" Q
.. Q:'$D(^VA(200,PERDUZ,203))
.. S STOP1=0
.. S XMNU=0 F S XMNU=$0(^VA(200,PERDUZ,203,XMNU))
Q:XMNU="!"('XMNU)!(STOP1=1) D
... I $G(^VA(200,PERDUZ,203,XMNU,0)) I
$P(^VA(200,PERDUZ,203,XMNU,0),"^",1)=OPTIEN S
STOP1=1,^TMP($J,"DVBA187P",PERDUZ,"USERSWITHOPTION")=""
;
;DOES THE USER HAVE ACCESS TO THE CURRENT KEY
S KEYNUM=$$LKUP^XPDKEY("DVBA CAPRI DENY GETVBADOCS")
I $G(KEYNUM)="" D BMES^XPDUTL("'DVBA CAPRI DENY GETVBADOCS' SECURITY
KEY HAS ALREADY BEEN DELETED. SECKEY^DVBA187P CAN NOT CONTINUE") Q
S PERDUZ=0 F S PERDUZ=$0(^VA(200,PERDUZ)) Q:PERDUZ="!"('PERDUZ) D
. I $D(^VA(200,PERDUZ,51,KEYNUM)) D Q
.. S ^TMP($J,"DVBA187P",PERDUZ,"DVBA CAPRI DENY GETVBADOCS")=""
.. I $D(^TMP($J,"DVBA187P",PERDUZ,"USERSWITHOPTION")) S
^TMP($J,"DVBA187P",PERDUZ,"USERSWITHOPTION")="DVBA CAPRI DENY GETVBADOCS"
;
```

Modified Logic (Changes are in bold)

```
DVBA187P(CONTINUED)
;ADD NEW SECURITY KEY TO ALL NON-TERMED PERSONS WHO DON'T HAVE OLD
KEY
    S KEYNUM=$$LKUP^XPDKEY("DVBA CAPRI GETVBADOCS")
    I $G(KEYNUM)="" D BMES^XPDUTL("'DVBA CAPRI GETVBADOCS' SECURITY KEY
HAS NOT BEEN INSTALLED ON SYSTEM. INSTALL AND RERUN SECKEY^DVBA187P") Q
    S PERDUZ=0 F S PERDUZ=$Q(^TMP($J,"DVBA187P",PERDUZ))
Q:PERDUZ="!"('PERDUZ) D
    . Q:$D(^VA(200,PERDUZ,3,"B","VISITOR"))=10 ;DO NOT INCLUDE USERS WHO
ARE VISITORS
    . Q:'$D(^TMP($J,"DVBA187P",PERDUZ,"USERSWITHOPTION"))
    . Q:$D(^TMP($J,"DVBA187P",PERDUZ,"TERMEDPERSON"))
    . Q:$D(^TMP($J,"DVBA187P",PERDUZ,"DVBA CAPRI DENY GETVBADOCS"))
    . ;IF AFTER FIRST RUN THIS ROUTINE IS RUN AGAIN EXCLUDE PERSON WITH
KEY FROM FIRST RUN
    . Q:$D(^XUSEC("DVBA CAPRI GETVBADOCS",PERDUZ))
    . K FDA,ERR,DIERR,KEYIEN
    . S FDA(200.051,"+1," PERDUZ ","",.01)=KEYNUM
    . S FDA(200.051,"+1," PERDUZ ","",1)=DUZ
    . S FDA(200.051,"+1," PERDUZ ","",2)=TODAY
    . S KEYIEN(1)=KEYNUM
    . D UPDATE^DIE("","FDA","KEYIEN","ERR")
    . S PERSON=$P(^VA(200,PERDUZ,0),"^",1)
    . I $D(DIERR) D BMES^XPDUTL("PERSON DUZ: " PERDUZ " (" PERSON ")
SHOULD BE ASSIGNED THE SECURITY KEY 'DVBA CAPRI GETVBADOCS' BUT COULD NOT IN
THE DVBA*2.7*187 POST INSTALL ROUTINE. PLEASE SET THIS PERSON MANUALLY") Q
    . D BMES^XPDUTL("PERSON DUZ: " PERDUZ " (" PERSON ") HAS BEEN
ASSIGNED THE NEW SECURITY KEY 'DVBA CAPRI GETVBADOCS'")
    ;
    K ^TMP($J,"DVBA187P")
    D BMES^XPDUTL("OK to delete DVBA197P")
    Q
    ;
;*****
*****
    ; AMIE EXAM (#396.6) file exam(s) to activate (create or update).
    ; Data should be in internal format.
    ; format: ien;exam name (60 chars);print name;body
system;routine;status;;wks#
;*****
*****
    ;
AMIENEW
    ;
    ;;463;DBQ Medical SHA;DBQ SEPRATN HEALTH ASMNT;1;DVBCQDRV;A; ; ;
    ;;QUIT
    ;
    Q
    ;
    ;
    ;
*****
    ; AMIE EXAM (#396.6) file exam(s) to rename. Data should be in
internal
```


Modified Logic (Changes are in bold)

```
DVBA187P (CONTINUED)
format.
        ; Format: ;;ien;"old" exam name(up to 60 chars);"new" exam name(up
to 6
0 chars)
        ;
        ;
*****
*****
EXOLDNEW ;
        ;;437;DBQ Medical Opinion 1;DBQ Medical Opinion
        ;;QUIT
        ;;
        Q

;*****
;*****
        ; AMIE EXAM (#396.6) file exam(s) to deactivate. Data should be in
        ; internal format.
        ; Format: ien;exam name (60 chars);

;*****
;*****
        ;
AMIEOLD  ;
        ;;416;DBQ MUSC Flatfoot (pes planus);16;DVBCQDRV;I;;
        ;;433;DBQ CARDIO Ischemic heart disease;6;DVBCQDRV;I;;
        ;;438;DBQ Medical Opinion 2;17;DVBCQDRV;I;;
        ;;439;DBQ Medical Opinion 3;17;DVBCQDRV;I;;
        ;;440;DBQ Medical Opinion 4;17;DVBCQDRV;I;;
        ;;441;DBQ Medical Opinion 5;17;DVBCQDRV;I;;
        ;;QUIT
        ;
        Q
```

- 6.2.2.13.2. Templates – Not applicable for CodeCR 544**
- 6.2.2.13.3. Bulletins– Not applicable for CodeCR 544**
- 6.2.2.13.4. Data Entries Affected by the Design – Not applicable for CodeCR 544**
- 6.2.2.13.5. Unique Record(s) – Not applicable for CodeCR 544**
- 6.2.2.13.6. File or Global Size Changes – Not applicable for CodeCR 544**
- 6.2.2.13.7. Mail Groups – Not applicable for CodeCR 544**
- 6.2.2.13.8. Security Keys – Not applicable for CodeCR 544**
- 6.2.2.13.9. Options – Not applicable for CodeCR 544**
- 6.2.2.13.10. Protocols– Not applicable for CodeCR 544**
- 6.2.2.13.11. Remote Procedure Call (RPC) Defined in Interface variables – Not applicable for CodeCR 544**
- 6.2.2.13.12. Constants Defined in Interface– Not applicable for CodeCR 544**
- 6.2.2.13.13. Variables Defined in Interface– Not applicable for CodeCR 544**
- 6.2.2.13.14. Types Defined in Interface – Not applicable for CodeCR 544**
- 6.2.2.13.15. GUI– Not applicable for CodeCR 544**
- 6.2.2.13.16. GUI Classes– Not applicable for CodeCR 544**
- 6.2.2.13.17. Current Form – Not applicable for CodeCR 544**
- 6.2.2.13.18. Modified Form– Not applicable for CodeCR 544**
- 6.2.2.13.19. Components on Form– Not applicable for CodeCR 544**
- 6.2.2.13.20. Events– Not applicable for CodeCR 544**
- 6.2.2.13.21. Methods– Not applicable for CodeCR 544**
- 6.2.2.13.22. Special References– Not applicable for CodeCR 544**
- 6.2.2.13.23. Class Events– Not applicable for CodeCR 544**
- 6.2.2.13.24. Class Methods– Not applicable for CodeCR 544**
- 6.2.2.13.25. Class Properties– Not applicable for CodeCR 544**
- 6.2.2.13.26. Uses Clause– Not applicable for CodeCR 544**
- 6.2.2.13.27. Forms– Not applicable for CodeCR 544**

6.2.2.13.28. Functions

6.2.2.13.29. Dialog

6.2.2.13.30. Help Frame

6.2.2.13.31. HL7 Application Parameter

6.2.2.13.32. HL7 Logical Link

6.2.2.13.33. COTS Interface

6.3. Network Detailed Design

CAPRI GUI communicates with various servers only within the VA intranet via TCP/IP. Existing VA intranet WAN backbone is leveraged as needed by any software component or service within the application. Type of traffic over TCP/IP includes, RPCBroker, HTTP and HTTPS.

6.4. Service Oriented Architecture / ESS Detailed Design

N/A

6.4.1. Service Description for <Consumed Service Name>

N/A

6.4.2. Service Design for <Provided Service Name>

N/A

7. External System Interface Design

This section details interfaces external to system, that are NOT services (ESS/SOA). Typically, these may include, RPCs, Flat Data Files etc.

7.1. Interface Architecture



ID	Name	Description	Interface type	Specifications
----	------	-------------	----------------	----------------

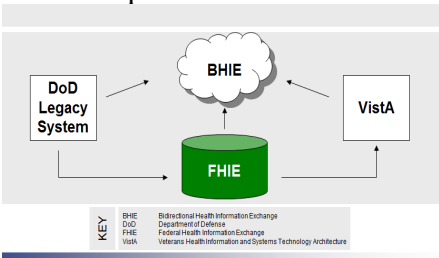
ID	Name	Description	Interface type	Specifications
7-1	<u>LOCAL VAMC</u> VISTA(CACHE) FILEMAN AND REMOTE PROCEDURE CALL/BROKER	InterSystems Caché is a commercial object-oriented database management system from InterSystems, used to develop healthcare management and telecommunications software. Customer software can use the database with object and SQL code. Caché also allows developers to directly manipulate its underlying data structures: hierarchical arrays known as M technology. FileMan is a set of utilities to provide a meta-data function for MUMPS applications. The FileMan utilities allow the definition of data structures, menus and security, reports, and forms, allowing someone to set up applications without tremendous experience in the MUMPS programming language. Remote Procedure Calls are a set of fileman instructions to bridge the gap between the GUI and VISTA. The RPC Broker is the foundation of a powerful set of RAD Delphi and Java components.	Data Base Managemen t	Cache20111 OpenVMS,Linu x Kernel 8.0 is required to run the RPC Broker and appropriate VISTA Software patch updates.
7-2	VISTA WEB SERVER	Veterans Health Information Systems and Technology Architecture (VistA) VistAWeb is an intranet web application used to review remote patient information found in VistA, the Bi-Directional Health Information Exchange (BHIE) system, the Health Data Repository II (HDR II) databases, and the Nationwide Health Information Network (NwHIN). (see http://healthit.hhs.gov/portal/server.pt?open_512&mode_2&cached_t rue&objID_1142).	Web Server	VistAWeb Version 16.1 (Patch WEBV*1*26)

ID	Name	Description	Interface type	Specifications
7-3	VIRTUAL VA	<p>Virtual VA Paperless Claims Processing is a Web-based suite of information solutions that provides electronic "eFolders" for claims processing through imaging, document management technologies, and integration with output capabilities of several other VA systems.</p> <p>Full implementation of Virtual VA enables VBA to address timely processing of services to veterans, which is currently hindered by the dependence on paper documents. Full implementation will also greatly improve the security and the privacy of veteran data by migrating from a vulnerable, paper claims folder to a secure, centralized, and electronic claims folder for compensation and pension processing.</p> <p>Electronic document workflow processing, business process management, VA and DoD systems integration,national scanning of documents, and improved workflow management.</p> <p>Currently, Virtual VA is designed to provide an electronic document management system for Compensation and Pension (C&P) Service and only for pension claims. Due to Congressional and Presidential interest in providing faster service to veterans, VA is proposing that Virtual VA be expanded to support paperless claims processing for all C&P activities. The web-based nature of the system will allow increased accessibility and information sharing across the 58 regional offices / external organizations.</p>	Virtual VA Web Service	Virtual VA-2009 [REDACTED] [REDACTED] [REDACTED]
CAPRI System Design Document		177		December 2014

ID	Name	Description	Interface type	Specifications
7-4	VETERANS BENEFIT MANAGEMENT SYSTEM	VBMS is an IT system designed to help claims adjudicators reach timely and informed decisions in a digital environment. VBMS will enable the five steps of the standard claims adjudication process including establishment, development, evidence, rating, and awards	Enterprise data services	External and inter-departmental communications , e.g. VRM, VLER Technology platform based on Services Oriented Architecture (SOA) principles that Is the foundation for future application development at VBA Incorporates legacy data and applications, e.g. VETSNET
7-5	WINDOWS BASED GUI(DELPHI)	CAPRI	Graphical User Interface	See Delphi Components and Extensions in this Document

ID	Name	Description	Interface type	Specifications
7-6	HeathleVet Web Services Client	As VistA is migrated to the new HealthVet-VistA system, some legacy VistA applications will need synchronous access to HealthVet application services and data. HWSC uses Caché's Web Services Client to invoke web service methods on external servers and retrieve results. It provides helper methods and classes to improve the use of the Web Service Client in a HealthVet-VistA environment	Web Server Manager Web Service Manager Lookup Key Manager	<p>HWSC acts as an adjunct to the web services client functionality provided in Caché, by:</p> <ul style="list-style-type: none"> -Leveraging Caché's platform-provided Web services client capabilities. -Adding a file and UI to manage the set of external web server endpoints (IP, port, etc.) -Adding a file and UI to register and manage the set of external web services. -Providing runtime API to invoke a specific web service on a specific web server. -Providing a runtime API to facilitate error processing in a VistA environment. -Providing a deployment API to install/register a web service proxy from a WSDL file. -Providing a management UI including the ability to 'ping' (test) a given web service/server combination

ID	Name	Description	Interface type	Specifications
7-6 cont				<p>-Providing a management UI including the ability to 'ping' (test) a given web service/server combination from VistA/M.</p> <p>Supporting both SOAP- and REST-style web services</p> <p>Fostering consistent implementation of VistA/M web service</p>
7-7	VENDOR LOCATIONS			
7-8	MULTIPLE VA ACCESS	<p>Links to multiple VAMCs from a drop down list to bring up that sites access logon prompt. Access credentials granted by the local ISO. The CAPRI Remote site selection screen displays the user's authorized VHA facilities. When the user selects a CAPRI Remote site executable it provides the authorized remote sites. If shown, the vertical scrollbar is used to scroll through all authorized sites. The user selects a site and then either double-clicks the site's name or clicks OK to access that site. CAPRI has been modified to include the city and state where each facility is located. In addition, the list may now be sorted by State.</p>	CAPRI GUI FORM/DROP DOWN LIST	<p>. (These accesses are established when an account is initially created and/or the user request specific facilities along with the proper approvals after the account creation. This facility information is located in the CAPRI file 396.96.)</p>

ID	Name	Description	Interface type	Specifications
7-9	CAPRI REMOTE USERS	Most VBA users are CAPRI Remote users. Each CAPRI Remote user needs only one Access Code and one Verify Code to connect to authorized VA Medical Center (VAMC) sites. CAPRI Remote users outside of VBA will normally obtain an access code from the Office of Information (OI) support staff, not from the local field site	Remote Access	The server for those users should be set to [REDACTED] [REDACTED] [REDACTED]
7-10	FHIE/BHIE FRAMEWORK	<p>Federal Health Information Exchange (FHIE)/ Supports monthly, 1-time transfer of electronic health information from DoD to VA at the point of a Service member's separation.</p> <p>Bidirectional Health Information Exchange (BHIE) Enables real-time sharing of electronic health information between DoD and VA for shared patients</p>  <p>The diagram illustrates the FHIE/BHIE Framework. It shows a central cloud labeled 'BHIE' connected to a 'DoD Legacy System' and 'VistA'. Below the cloud is a green cylinder labeled 'FHIE'. Arrows indicate data flow between the DoD Legacy System, BHIE, VistA, and FHIE. A key at the bottom identifies the components: BHIE (Bidirectional Health Information Exchange), DoD (Department of Defense), FHIE (Federal Health Information Exchange), and VistA (Veterans Health Information and Systems Technology Architecture).</p>	<p><i>FHIE One-way, enterprise exchange of text data</i></p> <p><i>BHIE Two-way, BHIE sites and all of VA exchange of text data</i></p>	<p>FHIE: Over 3 million unique patients</p> <p>Over 45 million lab results</p> <p>Over 7 million radiology reports</p> <p>Over 45 million pharmacy records</p> <p>Over 41 million standard ambulatory data records</p> <p>Over 2 million separated Service members have presented to the VA</p> <p>~2,700 messages transmitted daily in VA patients treated by DoD under local sharing agreements</p>

ID	Name	Description	Interface type	Specifications
7-11	DEPARTMENT OF DEFENSE	<p>The DoD and the VA have been working together, starting in 2002, to share health information between their two Electronic Health Record (EHR) systems. The VA's EHR is Veterans Health Information Systems and Technology Architecture / Computerized Patient Record System (Vista CPRS) and the DoD's is Armed Forces Health Longitudinal Technology Application (AHLTA), formerly CHCS II, US DoD military health system. Transferring information between Vista CPRS and AHLTA was a major undertaking, since both EHRs were created using different types of software applications.</p> <p>Both the DoD and the VA have been committed to the task and worked together to develop advanced technology to ensure the secure transfer of electronic health information between the two EHR systems. This important work supports the health care provided to active duty service members and veterans</p>	Remote Information transfer	When a capri patient is selected the list of facilities care was provided will include Station 200 if DOD Data is available. Station 200 queries the BHIE framework for needed data element for presentation to the connected VA in the selected report type.

ID	Name	Description	Interface type	Specifications
7-12	VLER/DAS JAVA AND DELPHI	This capability with the eHealth Exchange renders VistAWeb a key component of Virtual Lifetime Electronic Record (VLER) electronic health information exchange. Uses Object Oriented Programming to access Vista Patient data.	Web Service	<p>VLER/DAS web service requires two-way SSL (also known as mutual authentication) . CAPRI has been modified to relay VLER DAS web service requests through the CAPRI Proxy service using one-way SSL.</p> <pre> libeay32.dll VLER/DAS SSL dynamically linked library - ssleay32.dll VLER/DAS SSL dynamically linked library </pre>

Delphi components and extensions

File extension	Type	Delphi Version	Description
.bdsdeploy	?		Deployment file
.bdsgroup	text/xml	Delphi BDS 2006 and earlier	BDS Project Group, combines several projects into a group. Only put under source control if you actually use that feature.
.bdsproj	text/xml	Delphi BDS 2006 and earlier	Borland Developer Studio Project File . Successor of the .dof file holding compiler options etc. Also used for opening a project.
.bpg	text	Delphi 7 and older	Borland Project Group, combines several projects into a group. Only put under source control if you actually use that feature.
.bpl	Binary	Delphi 4 and newer	Borland Package Library . DLL including VCL components available for use in the Delphi environment at design time or by applications at run time.
.cbk	Other		Temporary Backup File .

.cfg	text	Delphi 7 and older	Project Configuration File. The Delphi IDE (up to Version 7) automatically creates this file from the settings in the .DOF file. It is only used by the command line compiler. See command line compilation .
.config			Config files contain project option information and build logic . Each project has a .config file.
.d	text		Output of the project unit dependencies. This is generated by checking the "Output unit dependency information" compile option or passing "-depends" dcc32.exe.
.dcl	binary		Delphi Component Library
.dep	Binary		Delphi Component Package . Contains symbol information for code compiled into a package. Used if building with runtime packages.
.dcpil	Binary		Delphi Compiler Symbolic Information . The purpose of a .DCPIL file is to provide symbolic information to the compiler about certain language constructs that can't be reconstituted from CLR metadata alone. This is a traditional Delphi technique for handling and caching compiler symbols.
.der	Resource		Delphi Component Resource File. Contains resource information for components, such as the icon to display in the palette.
.dcu	Binary	Win32	Delphi Compiled Unit File.
.dcuil	Binary	.NET	Delphi Compiled Unit File
.ddp	Other	Delphi 7 and older	Delphi Diagram Portfolio File. Used by the diagram editor.
.dfm	binary or text		Delphi Win32 Form File. A file listing the various objects and their property values contained within a form, including such things as form control coordinates. Right click on the form and select View as Text from the pop-up menu to view this file. Manual alterations to this file may prevent the IDE from being able to load the form. Use the Environment Options dialog to specify which format to use for newly created forms. The text format is recommended as it better supports source control.
.dof	text	Delphi 7 and older	Delphi Project Options File. Text file containing project options (such as compiler and linker settings, version info, search path and output directories).
.dpc	?		Delphi Package Collection
.dpk	text		Delphi Package Project
.dpkw	text		Delphi Package Project for Windows (Never seen in the wild)
.dpl	Binary	Delphi 3 and older	Delphi Package Library. Superseded by .bpl in later Delphi versions.
.dpr	text	All Delphi versions	Delphi Project File . The main file for any project. Contains the source code of the main file for a Delphi project. It is the primary entry point for the executable and refers to the other source files in the project. Do not delete.
.dproj	text/xml	Delphi	Delphi Project File for the native compiler . Contains the current

		2007 and newer	settings for project options, such as compiler and linker settings, directories, conditional directives, and command-line parameters. Set these options using Project > Options. Replaces bdsproj file.
.drc	text		Resource String File. Contains resource string information. Can be deleted; will be regenerated from source code by compiler.
.dres			Delphi compiled resource file .
.dsk	text		Delphi Desktop File . Contains information about the IDE desktop settings, including which files are open and the position of windows.
.dsm	Binary	Delphi 6?	Browser Symbol information from the last successful compile. Enviroment -> Preferences -> Desktop contents.
.dst			File used to save the desktop speed setting as set in the IDE toolbar desktop combo box.
.groupproj	text/xml	Delphi 2007 and newer	Project Group File. Replaces .bdsgroup file.
.identcache	Binary		Temporary Cache File created by refactoring engine to improve performance.
.int	text		Interface part of Unit not shipped with source, typically in \$(Delphi)\doc
.local	text/xml	Delphi 2005 and newer	User-specific project options . When using Starteam projects it will contain overrides to settings from the .bdsproj file.
.map	text		The optional error map file will contain a list of error addresses and source line numbers, grouped together for each source file in the application. To create a detailed .map file, ensure the {D+} compiler directive is set, and the "Detailed Map File" option selected in the Project >Options>Linker page of the IDE.
.mts	QA Metrics set file	Rad Studio 2010 enterprise or newer	The stored settings for QA metric
.nfm	Resource	Delphi 8.0 or newer	Delphi .NET Form File.
.pas	Source		Delphi Unit Source File.
.proj			Codegear RAD Studio project.
.res	Binary		<i>Binary</i> . Resource File . Binary file associated with a project containing resource definitions, such as strings, icons, images etc.This file will be recreated by the IDE when loading a project but the icon will get lost in that process. Therefore you should either keep it in source control or find another way to create it.
.resources	Binary		A binary resource file that can be embedded in a runtime executable.
.rsm	Binary		Remote Debugging Symbols file.

.tlb	Binary		Type library.
.todo			The project to-do list.
.tvconfig	text/xml		Modeling configuration file.
txvpck, txvcls			Information for model diagram.
.vlb	text/ini		Visual Live Bindings

7.2. Interface Detailed Design

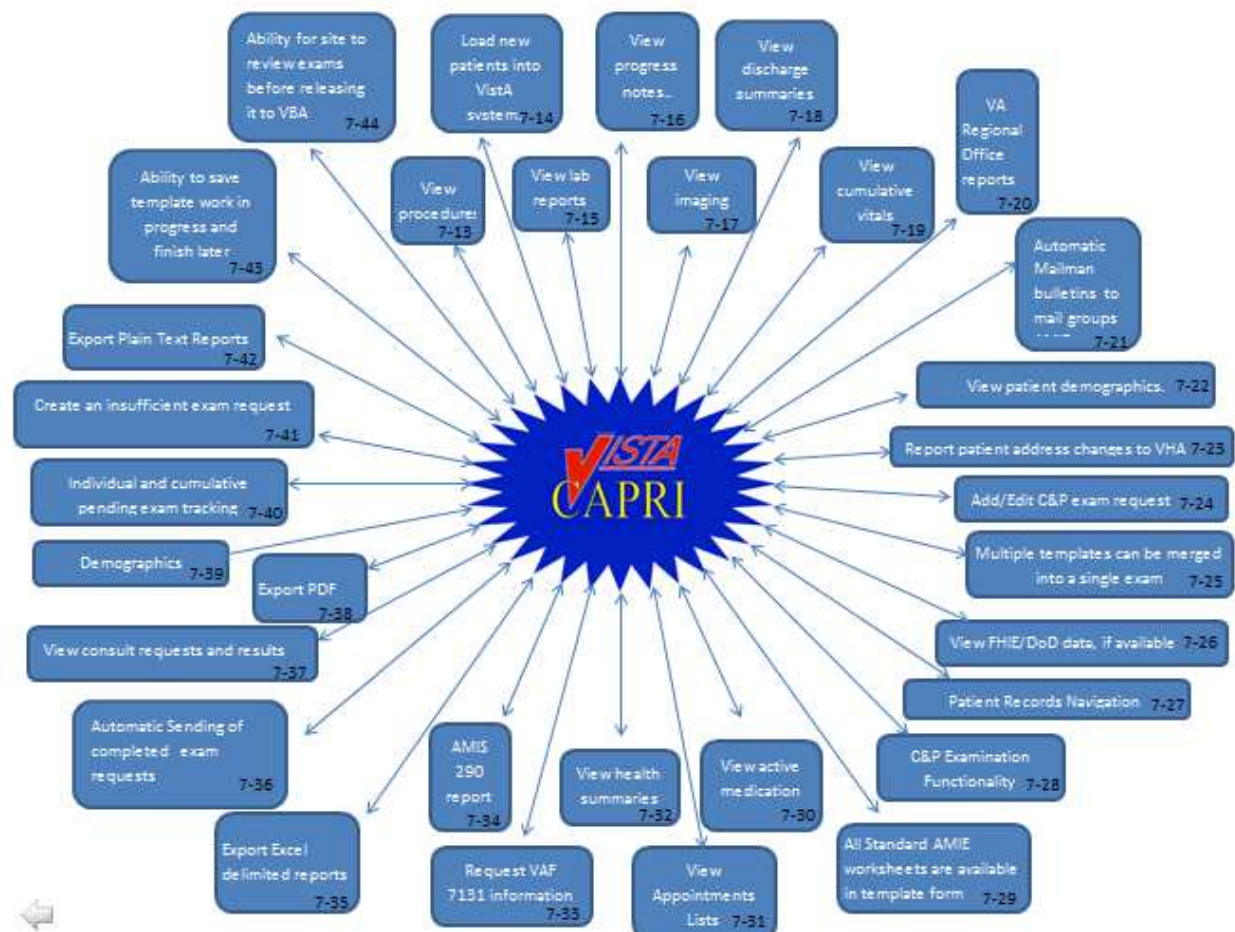


Table 7-2

ID	Interface Name	Description	Interface type
----	----------------	-------------	----------------

ID	Interface Name	Description	Interface type
7-13	View Procedures	When the user selects the clinical documents tab another complete set of tabs opens along the bottom of the screen Selecting the Clinical Documents tab yields access to View Procedures(Holter Monitor, Echocardiogram, Electrocardiogram, Thallium Stress Test, etc.)	CAPRI GUI Clinical Documents Tab
7-14	Load new patients into VistA system	VBA users should only establish a new patient within the VistA system if it is necessary to request a C&P examination for a veteran who is not a current patient within the medical facility's database. If the patient NAME and SSN search using CAPRI's Patient Selector function yields no results, i.e., "Match not found," then the user must establish the veteran as a new patient	CAPRI GUI Patient Selector Form
7-15	View lab reports	When the user selects the clinical documents tab another complete set of tabs opens along the bottom of the screen Selecting the Clinical Documents tab yields access to Laboratory Findings	CAPRI GUI Reports selection Form/Tab
7-16	View progress notes..	Notes Tab This tab allows the user to view or print the veteran's progress notes (Error! Reference source not found.). The left side of the screen shows the appointment date, type of progress note, and the author of the note. If the user places the mouse pointer over a progress note, the full title is displayed. When a note is selected, it appears in the window on the right.	CAPRI GUI Notes Tab
7-17	View imaging	When the user selects the clinical documents tab another complete set of tabs opens along the bottom of the screen Selecting the Clinical Documents tab yields access to Imaging (X-rays, CT, MRI, etc.	CAPRI GUI Clinical Documents Tab
7-18	View discharge summaries	When the user selects the clinical documents tab another complete set of tabs opens along the bottom of the screen Selecting the Clinical Documents tab yields access to Discharge Summaries	

ID	Interface Name	Description	Interface type
7-19	View cumulative vitals	<p>This creates a report that matches the data generated by the AMIE option of the same name. It is a cumulative report containing all admissions for a given date range. The report is designed primarily as an auditing tool for the RO. Information which may be provided for each patient includes claim number, claim folder location, SSN, admission date, admitting diagnosis, discharge date, bed service, whether the patient is receiving A&A or pension, and eligibility data. Depending on the date range selected, this report can be quite lengthy.</p> <p>Step 1 The user enters a date range using Start Date and Stop Date.</p> <p>Step 2 A specific regional office is selected.</p> <p>Step 3 – The user clicks OK</p>	CAPRI GUI Reports selection Form/Tab
7-20	VA Regional Office reports	<p>AMIS 290 This option electronically produces the AMIS C&P report that was manually produced by the regional office. It is a general system of computer programs used to process management reports. The AMIS 290 report covers compensation and pension examination request activity. The regional office AMIS 290 calculates the data based only on that specific regional office's requests, or when the regional office is left blank, it calculates the data for all regional offices</p>	CAPRI GUI REPORTS Form

ID	Interface Name	Description	Interface type
7-21	Automatic Mailman bulletins to mail groups AMIE	<p>AMIS 290, Processing time for an insufficient request includes the processing time of the original request. In addition to a hard copy being produced, this option allows the user to send a MailMan message either locally or via network mail. The mail bulletin contains the same information that appears on the report. Vocational Rehabilitation Tab- Vocational Rehabilitation and Employment (VR&E) employees require the ability to refer a veteran to a medical center for the purpose of specific types of exams or opinions in order to determine VR&E benefits.</p> <p>This feature enables users to create an electronic request for medical services, track those requests and print reports. A consult is created outside the CAPRI application. The request can be linked to the consult by a VHA Coordinator. Notifications are sent to both VR&E Counselor (via Outlook) and VHA Coordinator (via MAILMAN) when the 28-8861 form status is updated. A nightly background job monitors consults that are linked to the 28-8861 file. Specific Events Trigger Mailman Messages; Upon specific events, MAILMAN messages are sent to VHA Coordinators who must be a member of the</p>	C&P Exams Reports, AMIS 290; Vocational Rehabilitation Tab; Specific Events

ID	Interface Name	Description	Interface type
7-21 contin ued	Automatic Mailman bulletins to mail groups AMIE (Continued)	<p>MAILMAN mail group: DVBA VR VOCREHAB PERSONNEL and the VR&E Counselor(s) receive email notifications via Outlook. These notifications are sent for the following status changes to the Medical Services Request:</p> <p>-New Status: This message reads: Chapter 31 Referral for Medical Services New. **This notification is only sent to the VHA Coordinator via MAILMAN</p> <p>-Pending Status: This message reads: Chapter 31 Referral for Medical Services Pending. **In CAPRI, once the VHA Coordinator/or designated person links the consult request to the original Chapter 31, VA Form 28-8861 request, the status of the consult in CAPRI is then changed to Pending. Both the VHA Coordinator and VR&E Counselor receive notification messages</p> <p>-Completed Status: This message reads: Chapter 31 Referral for Medical Services Completed ** Both the VHA Coordinator and VR&E Counselor receive notification messages</p> <p>-Cancelled Status: This message reads: Chapter 31 Referral for Medical Services Cancelled** Both the VHA Coordinator and VR&E Counselor receive notification messages</p>	
7-22	View patient demographics.	Individual Patient Demographics can edited after selecting the patient from the CAPRI Patient Selector Form	CAPRI GUI CAPRI Patient Selector Form

ID	Interface Name	Description	Interface type
7-23	Report patient address changes to VHA	<p>Address Verification Form; CAPRI repeatedly prompts the user to check the latest address of record in the VistA system (Error! Reference source not found.). If the veteran has a new address, the user clicks the Edit Address Now button. If the user is not sure if the new address is more current than the one shown by CAPRI, then the Permanent Address is not updated and instead this fact is noted and the new address is included in the general comments area of the exam request.</p> <p>If the Permanent Address: shown by CAPRI is current, the user clicks OK. If the user enters a change to the Permanent Address: information, it does not update the address information directly in the VistA Registration database; instead this information is appended as text to the examination request, where it can be further edited if necessary.</p>	CAPRI GUI Form Address Validation Form
7-24	Add/Edit C&P exam request	<p>CAPRI prompts the user to check the latest address of record in the VistA system (Error! Reference source not found.). If the veteran has a new address, the user clicks the Edit Address Now button. If the user is not sure if the new address is more current than the one shown by CAPRI, then the Permanent Address is not updated and instead this fact is noted and the new address is included in the general comments area of the exam request.</p> <p>If the Permanent Address: shown by CAPRI is current, the user clicks OK. If the user enters a change to the Permanent Address: information, it does not update the address information directly in the VistA Registration database; instead this information is appended as text to the examination request, where it can be further edited if necessary.</p>	CAPRI GUI Add/Edit C&P Exam request form

ID	Interface Name	Description	Interface type
7-25	Multiple templates can be merged into a single exam\	<p>CAPRI prompts the user to check the latest address of record in the VistA system (Error! Reference source not found.). If the veteran has a new address, the user clicks the Edit Address Now button. If the user is not sure if the new address is more current than the one shown by CAPRI, then the Permanent Address is not updated and instead this fact is noted and the new address is included in the general comments area of the exam request.</p> <p>If the Permanent Address: shown by CAPRI is current, the user clicks OK. If the user enters a change to the Permanent Address: information, it does not update the address information directly in the VistA Registration database; instead this information is appended as text to the examination request, where it can be further edited if necessary.</p>	CAPRI GUI Browse Templates Form
7-26	View FHIE/DoD data, if available	<p>CAPRI Remote users have access to the DoD Tab. From this tab, DoD records are available for certain veterans beginning approximately six weeks after discharge. These records are available through the Federal Health Information Exchange (FHIE). The user knows if DoD records are available when they access the Patient Selector screen and the notation DOD data is available indicates that the user can access these records after selecting the patient. The patient demographic data is not visible in this specific example because it is a sensitive level record.</p>	CAPRI GUI Tabbed Form DoD Records Tab
7-27	Patient Records Navigation	<p>Patent Selector Screen Local and Remote Sites give the ability to Navigate patient records to get a more complete understanding of care</p>	CAPRI GUI Patient Selector Form

ID	Interface Name	Description	Interface type
7-28	C&P Examination Functionality	<p>C&P Exam Requests</p> <p>The C&P Exam tab (Figure 8-) includes functions such as:</p> <ul style="list-style-type: none"> -Requesting C&P examinations -Viewing/editing requests -Canceling requests -Adding an exam to a pending request -Creating status inquiry reports -Viewing completed requests -Tracking the progress of the request for claims management purposes -Requesting an Insufficient Exam- -Printing results for individual patients <p>Pending requests are shown with a date only in the left column. Completed requests have a completion date in the right column.</p> <p>When the user first accesses this screen, none of the examinations are selected, and only the Add a New Request button is enabled. After the user selects an examination, the Re-Print Final C&P Results, Status Inquiry, and View/Edit Selected Request buttons are enabled.</p>	CAPRI GUI C&P Exam Request Form
7-29	All Standard AMIE worksheets are available in template form	Selecting AMIE Worksheets takes the user to a C&P website that provides Index to Disability Examination Worksheets. Selecting a worksheet displays instructions for using the worksheets	CAPRI GUI Tools Tab/ Web Services ;
7-30	View active medication	When the user selects the clinical documents tab another complete set of tabs opens along the bottom of the screen Selecting the Clinical Documents tab yields access to View Active Medications	CAPRI GUI Reports Selection Form/Tab
7-31	View Appointments Lists	<p>Patient Profile Medical administration Service (MAS) (Full)</p> <p>The following screen is the default – it specifies all dates, appointments, enrollments, team information, edits, dispositions, and the means test. The user can change the generated report to exclude particular types of information by selecting No for that particular type</p>	CAPRI GUI Reports Selection Form/Tab

ID	Interface Name	Description	Interface type
7-32	View health summaries	Health Summaries are customized reports comprising VistA components specified by end users. Most of these summaries were developed by the VHA facilities. Regional Offices can create special summaries that appear on the menu. The Veterans Administration Regional Office (VARO) St. Petersburg, in cooperation with the Veterans Integrated Service Network (VISN) 8, developed a Health Summary called VARO Rating, which contains components specified by RVSRs to facilitate their work process. To develop a custom Health Summary, the user must contact the VHA facility's IRM. The VARO Rating Health Summary in VISN 8 facilities include the following VistA components: demographics, imaging impressions, past and future clinic appointments, admissions and discharges, discharge summaries, progress notes, surgery reports, and medications. In addition to the reports on the menu, the user can create one-time Ad Hoc reports for use with a particular case	CAPR GUI Health Summaries Tab
7-33	Request VAF 7131 information	7131 requests can be made for reports including: -Patient records which may be retired after a long period of facility inactivity -Patient records which only exist on paper -VAF 21-2680 Aid and Attendance examinations that have been completed by the veteran's health care provide;Competency reports;Asset information;21-day certificates;Records based upon hospital admissions such as discharge notices and discharge summaries	CAPRI GUI 7131 Request tab
7-34	AMIS 290 report	The AMIS 290 report covers compensation and pension examination request activity. By selecting one of the following filters to run the AMIS 290 report, CAPRI produces separate statistics as indicated: -Agent Orange -Benefits Delivery at Discharge and Quick Start -IDES -All priorities except AO, BDD, IDES , and QS	CAPR GUI Reports Selection Form/Tab

ID	Interface Name	Description	Interface type
7-35	Export Excel delimited reports	Reports options The data can be downloaded from these reports in text delimited or comma separated value file formats so that further analysis can be performed on the data such as searching and sorting; where these capabilities are provided by external applications capable of accepting the data (e.g.: Microsoft Access® or Microsoft Excel®).	CAPR GUI Reports Selection Form/Tab
7-36	Automatic Sending of completed exam requests	When an examiner completes an exam from the signature validation screen it will automatically be transmitted to Virtual VA. The examiner will need to acknowledge the Virtual VA Transmission Status window as shown below by clicking the “Close” button	
7-37	View consult requests and results	Notification Messages: CAPRI, once the VHA Coordinator/or designated person links the consult request to the original Chapter 31, VA Form 28-8861 request, the status of the consult in CAPRI is then changed to Pending . Both the VHA Coordinator and VR&E Counselor receive notification messages	
7-38	Export PDF	Documents handled by CAPRI can be in Portable Document Format	

ID	Interface Name	Description	Interface type
7-39	Demographics	<p>View Registration Data Report</p> <p>This report provides full demographic data, including military information, for the selected patient.</p> <p>Health Summaries Tab; Health Summaries are customized reports comprising VistA components specified by end users. Most of these summaries were developed by the VHA facilities. Regional Offices can create special summaries that appear on the menu. The Veterans Administration Regional Office (VARO) St. Petersburg, in cooperation with the Veterans Integrated Service Network (VISN) 8, developed a Health Summary called VARO Rating, which contains components specified by RVSRs to facilitate their work process. To develop a custom Health Summary, the user must contact the VHA facility's IRM. The VARO Rating Health Summary in VISN 8 facilities include the following VistA components: demographics, imaging impressions, past and future clinic appointments, admissions and discharges, discharge summaries, progress notes, surgery reports, and medications. In addition to the reports on the menu, the user can create one-time Ad Hoc reports for use with a particular case.</p>	CAPR GUI Reports Selection Form/Tab; Health Summaries Tab;
7-40	Individual and cumulative pending exam tracking	<p>C&P Examination Functionality</p> <p>CAPRI automatically checks pending Compensation and Pension Worksheet Module (CPWM) Template statuses. Pending templates in the user's queue are displayed on the alert screen. The C&P Alert screen displays alerts according to template status. Alerts for template statuses are: draft, awaiting signature, sent back from reviewer, requiring review, CPRS documents to cosign, and cosigned documents ready to transfer to AMIE.</p>	CAPRI Alerts

ID	Interface Name	Description	Interface type
7-41	Create an insufficient exam request	Add An Insufficient Exam Request if this examination was completed but is insufficient This option is used if a completed examination is not sufficient, and the RO has followed local procedures to attempt to make the examination sufficient for rating purposes, The user verifies that the correct request was selected, and then clicks the Add An Insufficient Exam Request button. – The Add New C&P Exam screen opens (Error! Reference source not found.). The user completes this screen as directed in Section Error! Reference source not found. , Add a New Request . The only difference is that the list of available examinations is limited to those completed in the previous, insufficient examination (Error! Reference source not found.). These choices only appear after the user enters the Routing Location	CAPRI GUI View C&P Exam “Edit” form
7-42	Export Plain Text Reports	Reports options The data can be downloaded from these reports in text delimited	CAPRI GUI Reports Selection Form
7-43	Ability to save template work in progress and finish later	A details bar is available on each free text area of the template. The details bar is designed to provide users with a larger data entry area. The details bar becomes visible when the mouse is over the text box. After the user clicks the details bar, a larger window is displayed. The user can then type the information desired and save and close the details box. When the details box is closed, the information is displayed in the original free text area	Save through RPC from GUI to VISTA

ID	Interface Name	Description	Interface type
7-44	Ability for site to review exams before releasing it to VBA	<p>Some sites require the exam to be sent to a reviewer when it is complete. One and only one of the following <u>Security Keys</u> must be assigned for anyone who is creating new C&P exam templates with CPWM.</p> <ol style="list-style-type: none"> 1. DVBAB CPWM DISALLOW REVIEW – User does not need documents reviewed prior to release. 2. DVBAB CPWM OPTIONAL REVIEW – User can choose to send some documents for review and not other documents. 3. DVBAB CPWM REQUIRE REVIEW – User must have all documents reviewed by a reviewer prior to upload. <p>OPTIONAL Security Key If the site chooses to use the review process, users designated as reviewers must be assigned the DVBAB CPWM REVIEWER Security Key</p>	VISTA Option and Key assignment

8. Human-Machine Interface

CAPRI provides VBA employees with a standardized, user-friendly method to rapidly access veterans' electronic medical records throughout the Department of Veterans Affairs (VA). CAPRI delivers leading edge "point and click" technology to the users' desktops. In addition, the learning curve for CAPRI is significantly less than that for character-based systems.

HUMAN		HARDWARE		SOFTWARE		
Brain	Eyes Ears Hands	Peripheral Hardware	Integrated Circuits	Kernel space	User Space	
		Display Monitor, Speakers	← Output	Video Subsystem. Audio Subsystem	VA Network, VistA, CAPRI Application	→ Input-Output Loop ←
		Keyboard, Mouse	Input →	Input Subsystem	^	

8.1. Interface Design Rules

CAPRI GUI follows standard windows application design conventions.

The GUI utilizes standard, 508 accessible, screen controls in all it's forms and dialogs.

Most commonly used controls are as follows

- Tabbed Panels
- Labels
- Edit boxes
- Pulldown controls
- Buttons
- Menus (common elements like "File", "Help", "Help -> About")

8.2. Inputs

All user input into the CAPRI application is via the GUI. Access to various entry screens or menu options that allow opening of some screens may be restricted by VistA Security Keys as managed by Local IT/IRM.

Logging onto CAPRI

The information is found in the latest version of the [CAPRI GUI User Manual](#), see section “Logging On” at the following URL: [REDACTED]

VistA Terminal

The “Attachmate Reflections Secure Shell” application replaces the telnet window. CAPRI Remote users can launch a VistA Terminal session by selecting the VistA button to log into the local VistA system site they are assigned to. A dialog box is displayed when the VistA button is clicked that provides the user the ability to choose between connecting using the secure shell application or telnet. The default is set to secure shell application. Telnet is not an authorized method of connection.

***Note:** Local CAPRI users will not have access to the VistA Terminal from CAPRI and the VistA button will not be available.*

Using the Software Patient Selector Screen or Patient Entry / Selection

***Security Note:** The CAPRI application does not allow users to view their own personal patient records. If a user attempts to do so, CAPRI prevents access and alerts the Security Administrator at the VHA facility who takes established security violation actions. In addition, when selecting a patient who is a VA employee, CAPRI allows the user to do so only after the user agrees to Privacy Act Terms via a dialog box. An alert is sent to the Security Administrator who then inquires about the user’s business reasons for accessing those records.*

Selecting a Patient / Veteran

The Patient Selector screen allows the user to search the VistA database for patients who have records. Patients can be located using a Social Security Number (SSN), the last name initial plus the last four digits of the SSN, or by typing in the veteran’s last name and first name. Current users of AMIE II will recognize that these are the same search methods for that application.

If the user is looking for existing VHA medical records and the patient selection search yields no results, then there are no existing VHA medical records for that patient at the facility that was accessed. Verify that the user typed in the SSN correctly and that the user is logged into the correct VHA facility.

***Note 1:** To avoid displaying sensitive information regarding our patients and staff, the examples in this manual contain contrived names instead of real names. Patients and staff are referred to as CPRIPATIENT, ONE; PROVIDER, ONE; or USER, ONE. Likewise, real SSNs, real addresses, and other personal identifiers are not used.*

Step 1 – To begin the search, the user clicks **File/Select Patient** and enters the veteran’s information (Figure 2-1). For example, if the user wants to view the records of CPRIPATIENT, TWO (SSN: 666080315), then the following would be the valid search methods:

- SSN (preferred method): the user enters 666080315 and clicks Select
- Last name initial and last four of SSN: the user enters C0315 and clicks Select
- Name: the user enters CPRIPATIENT, TWO and clicks Select

Note 2: *There is NO spaces between the comma and the first name.*

Step 2 – CAPRI displays a list of possible matches (Figure 2-1). The user can click once on the patient's name to display more information to help verify that the user has selected the correct veteran. This additional information includes the patient's full name, SSN, claim number, gender, age, and date of birth (DOB). If there is only one match, this information is displayed automatically. After choosing the correct veteran, the user clicks the **Select** button.

Step 3 – After the user clicks **Select**, CAPRI builds all of the background information on the selected veteran and automatically opens the C&P Exam tab. From this starting point the user can request a C&P exam for the selected veteran or select any of the other tabs to navigate existing patient records.

The screenshot shows a window titled "Patient Selector". On the left, there is a list of patient names: CPRIPATIENT_FIVE, CPRIPATIENT_FOUR, CPRIPATIENT_ONE, CPRIPATIENT_SIX, CPRIPATIENT_THREE, and CPRIPATIENT_TWO. The name CPRIPATIENT_TWO is highlighted in blue. Above this list, the "Patient ID:" field contains the text "CPRI". To the right of the list, under the heading "Selected Patient", the following information is displayed: CPRIPATIENT_TWO, SSN: [REDACTED], MALE, [REDACTED], Ward: [REDACTED], Claim Number: [REDACTED], and ICN: [REDACTED]. Below this information is a button labeled "Other Facilities Visited". At the bottom of the window, there are several buttons: "Demographics", "More", "Enter New Pt.", "Cancel", "Enterprise Search", and "Select".

Figure 8-2

Figure 8-3 is the result screen after selecting a patient.

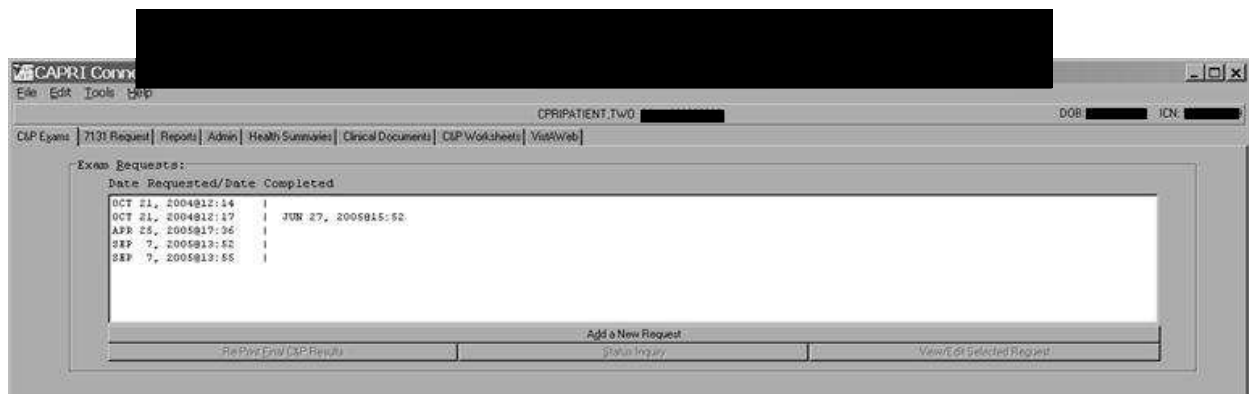


Figure 8-3

Step 4 – To select another patient, the user selects **File/Select Patient** (Figure 8-4) and returns to **Step 1** of this section.

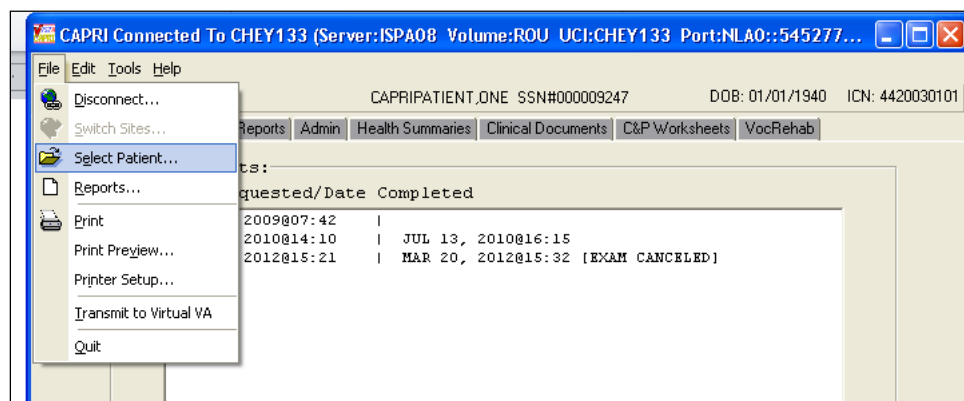


Figure 8-4

Note: CAPRI notifies the user if the patient is deceased and allows the user to Continue or Cancel. When the user searches using the Patient Selector screen, a message indicates if the patient is deceased (Figure 8-5).

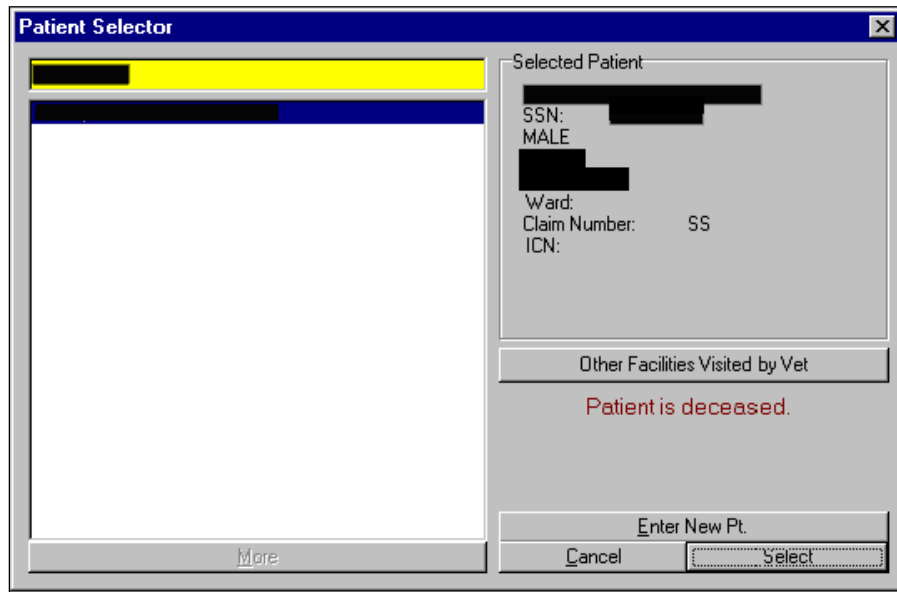


Figure 8-5

If the user clicks **Select**, the following dialog box appears (Figure 8-6) with the patient's date and time of death.

The user selects **Yes** to continue or **No** to cancel.

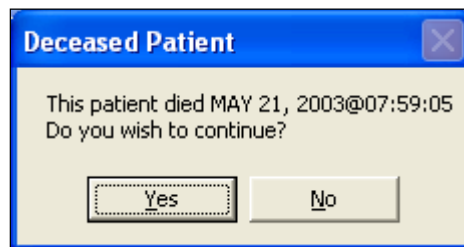


Figure 8-6

8.2.1 Sensitive Records

A user's own record and the records of the user's colleagues are designated as sensitive records. The user may not access them without justification. If a user clicks the name of patient whose record is sensitive, the words **Sensitive Record** (Figure 8-7) are displayed in place of the patient's personal information.

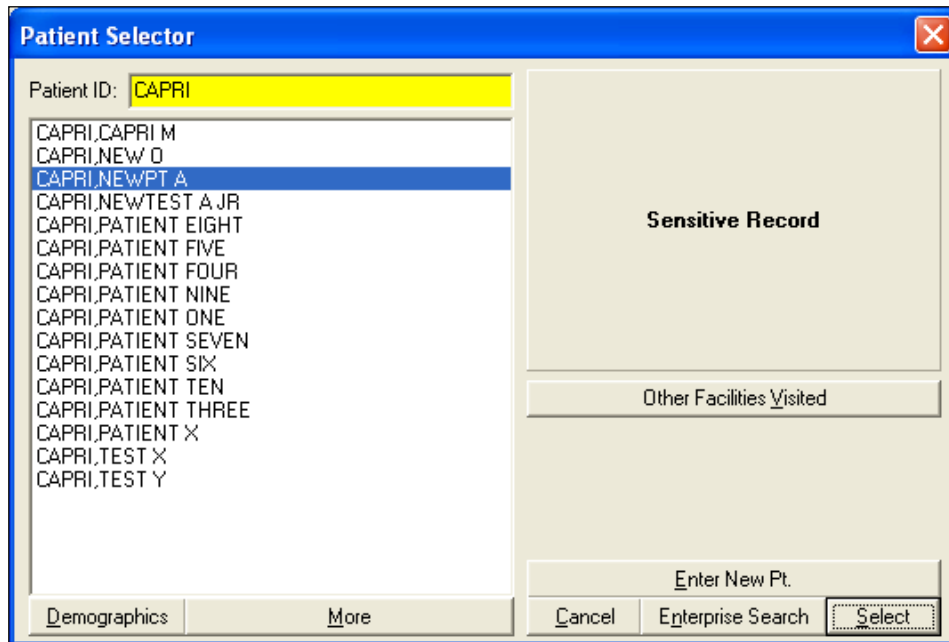


Figure 8-7

If the user attempts to open this record, a warning message is displayed (Figure 8-8).

If the user does not have justification for viewing this record, then the user should select **No** on this message box.

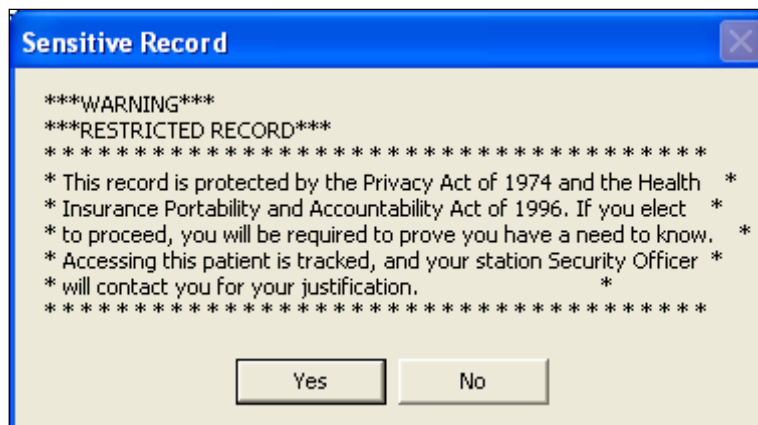


Figure 8-8

8.2.2 Entering a New Patient / Veteran

Note: VBA users should only establish a new patient within the Vista system if it is necessary to request a C&P examination for a veteran who is not a current patient within the medical facility's database. If the patient NAME and SSN search using CAPRI's Patient Selector function yields no results, i.e., "Match not found," then the user must establish the veteran as a new patient.

Step 1 – The user selects the **Enter New Pt.** button at the bottom of the Patient Selector screen (Figure 8-8).

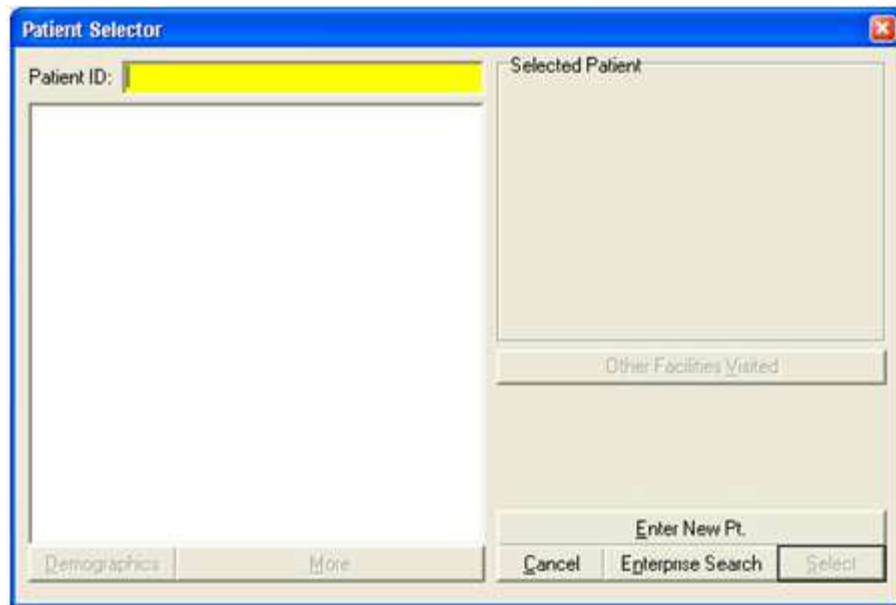


Figure 8-9

Steps 2 – User enters the veteran’s SSN in the space provided, and then clicks **Verify SSN is Not in Use** button (Figure 8-10).



Figure 8-10

Step 3 – If the SSN is not associated with an established patient record, CAPRI opens the **Enter New Patient** template (Figure 8-11).

When the user selects the **Click to Continue** button, a query is run to find any existing patient with the following information similar to the new patient:

- First two letters of the First and Last name are the same
- Same last four of the SSN
- Same Year and Month of the Birthdate
- Same Year and Day of the Birthdate

Enter New Patient

You may enter a new patient into the Vista patient file using this dialog.

SSN:

First Name: Gender:

Middle Name: DOB:

Last Name:

Jr., St., etc:

Figure 8-11

The query results are displayed so that the user can manually verify that the new patient does not exist in the system. The **name, gender, DOB, and SSN** of the potential matches are displayed, as shown in Figure 8-12.

Patient File Matches

Based on the information entered, there are potential matches.
Please verify that you are not creating a duplicate patient.

PATIENT, TEST A	MALE	<input type="text"/>
PATIENT, TEST C	MALE	<input type="text"/>
PATIENT, TEST NEWEMPL	FEMALE	<input type="text"/>
PATIENT, TEST QUALITY JR	MALE	<input type="text"/>

Figure 8-12

8.3. Outputs

The following are outputs from the CAPRI Software:

- C&P Exam Requests

- 7131 Request
- Reports (Patient Specific, Non Patient Specific, Consolidated Remote Reports)

C&P Exam Requests

The C&P Exam tab ([Figure 8-](#)) includes functions such as:

- Requesting C&P examinations
- Viewing/editing requests

8.3.1. View/Edit Selected Request

Step 1 The user logs into CAPRI, looks up the veteran using the **Patient Selector**, and moves to the **C&P Exams** tab ([Figure 8-3](#)). If there are no examination requests, the window is blank. If the veteran already has examination requests on file, the window shows the dates requested.

Step 2 The user selects the examination request date and clicks the **View/Edit Selected Request** button.

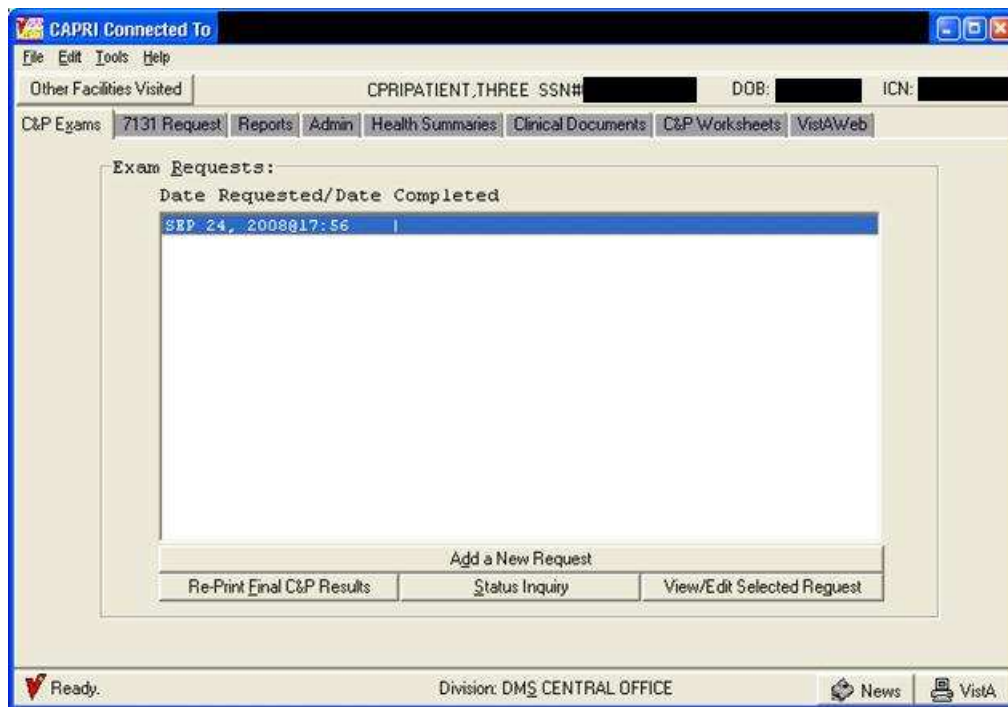


Figure 8-13

Step 3 – The **View C&P** screen opens, showing all of the entries from the original request, as well as the examination status ([Figure 8-14](#)). The user can scroll down to see additional information from the original request. The exam in this request cannot be edited because it is already complete, but the original request may be viewed.

View C&P Exam

Edit

Request Reference #: [REDACTED] Request Status: RELEASED TO RO, NOT PRINTED

Pt. Name: [REDACTED] Last Rating Exam Date: N/A

Claim Folder Required? YES Priority of Exam(s): IDES

Request Date: JUN 07, 2013@11: RO: CHEYENNE VAMC

Requested By: DELACRUZ,KRISTINAM Routing Location: CHEYENNE VAMROC

Comments:

CLAIMS FILE BEING SENT FOR REVIEW BY THE EXAMINER.

Exams Requested:

DBQ GI Intestines (other than surgical or infectious) [COMPLETE]
 DBQ GI Intestines (surgical) [COMPLETE]
 DBQ CARDIO Hypertension [COMPLETE]
 DBQ CARDIO Ischemic heart disease [COMPLETE]
 DBQ Cold injury residuals [COMPLETE]

Cancel ALL Exams View Selected Exam Add Exam to Request

Figure 8-14

Step 4 – If the user wants to edit a request that is still pending and has not yet been scheduled, it can be done on the **View C&P Exam** screen shown in [Figure 8-](#). Otherwise, skip to **Step 8**. The user can edit **Claim Folder Required?**, **Priority of Exam(s)**, **Routing Location**, **Comments**, and **Other Disabilities** by typing directly into those fields. To add another examination, the user clicks the **Add Exam to Request** button shown in [Figure 8-15](#).

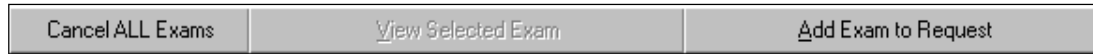


Figure 8-15

Step 5 – If the **Add Exam to Request** button is clicked, the exam listing field opens allowing the user to make a selection (16). The user can scroll down the list, or use the **Find by Body System** button. In the **Find by Body System** view ([Figure 8-15](#)), the user scrolls down the body systems until the correct one is found, then single-click the “+” box in front of the body system name, or the user can double-click the body system name to open a list of all pertinent examinations. The user double-clicks the desired examination to add it. The view reverts to the scroll list view, and that examination is checked.

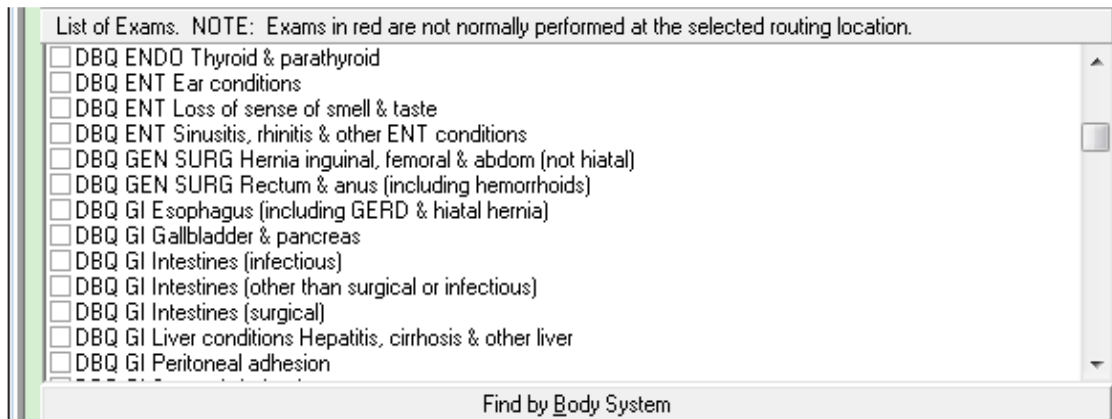


Figure 8-15

Step 6 – In the scroll list view ([Figure 8-17](#)), the user clicks the checkbox next to any of the desired examination.

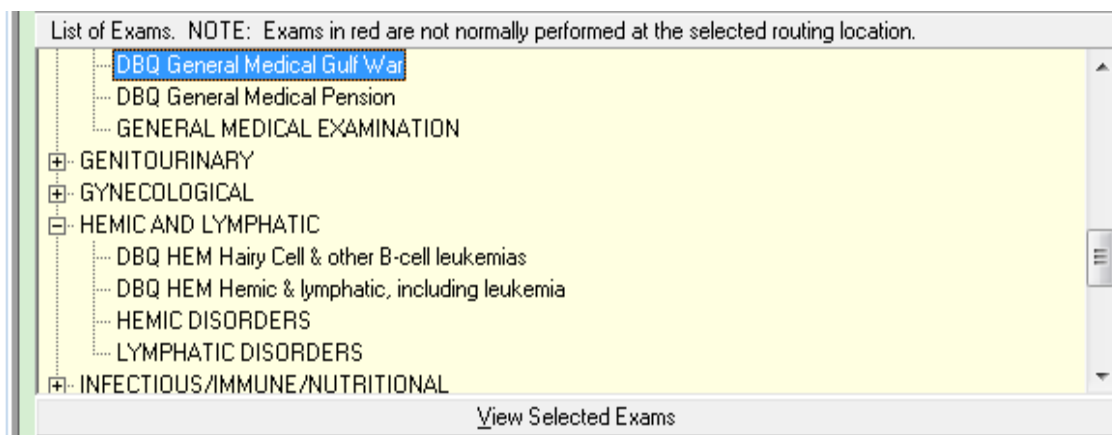


Figure 8-17

Step 7 The user selects **Add Selected Exams** or **Cancel Add Exams** as appropriate ([Figure 8-18](#)).

Step 8 The user can select **Close Window** to close this screen or **Add An Insufficient Exam Request** if this examination was completed but is insufficient ([Figure 8-19](#)). See the **Error! Reference source not found.** section for additional information.

7131 Request

The 7131 Request tab includes functions such as:

- Adding new requests for 21-Day Certificates, Notices of Discharge, etc...
- Status inquiries and reports
- Viewing and editing pending requests

Note: 7131 requests are reserved for information that cannot be obtained directly through CAPRI, such as older records and retired records that may not exist in the electronic database or other records in VHA's CPRS that are unavailable in CAPRI. For example, scanned records in CPRS may not be available in CAPRI.

Pending requests are shown in the left column and completed requests are shown in the right column of the screen. In the following example ([Figure 8-17](#)), the request is still pending for an Admission Report, and there are no completed requests. When the tab is first opened, all pending and completed requests are shown, but none is selected. The **Status Inquiry** and **View/Edit Selected Request** buttons are not available until a request is selected.

[Figure 8-20](#) shows what an open 7131 request for an outpatient treatment record looks like. The **Date of Event/Date Request Completed** displays **Out Patient (O/P) Activity** when the 7131 request is not for an Admission Report.

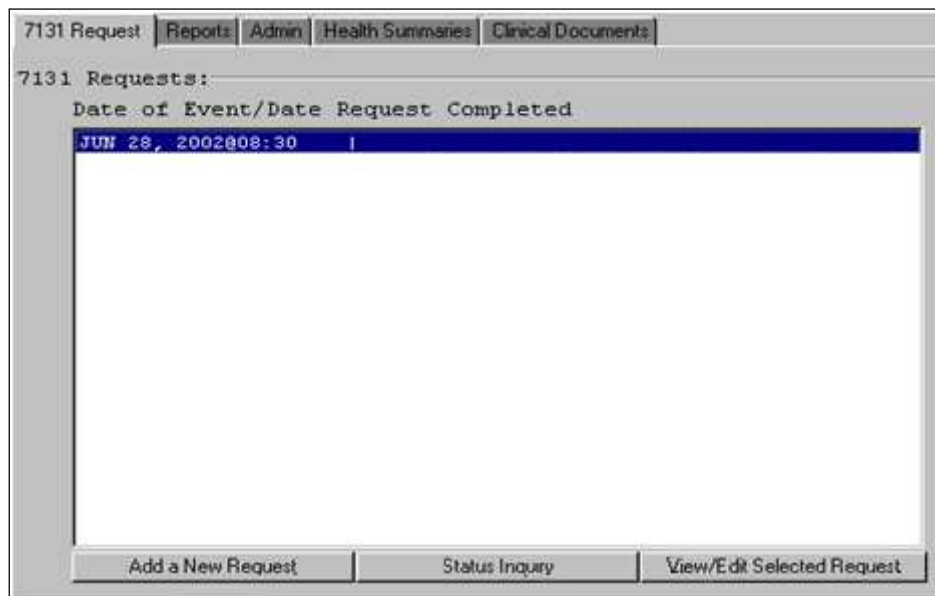


Figure 8-20

8.3.2. Add a New 7131 Request

7131 requests can be made for reports including:

- Patient records which may be retired after a long period of facility inactivity

- Patient records which only exist on paper
- VAF 21-2680 Aid and Attendance examinations that have been completed by the veteran's health care provider
- Competency reports
- Asset information
- 21-day certificates
- Records based upon hospital admissions such as discharge notices and discharge summaries

To add a new 7131 request, do the following:

Step 1 The user logs into CAPRI, selects a patient, and clicks the **7131 Request** tab.

Step 2 The user clicks the **Add a New Request** button ([Figure 8-](#)).

Step 3 – The user selects either **Outpatient/Activity** or **Admission** (Figure 8-22) depending on which one most closely relates to the request. If no admission is listed under the **Select Admission Date** (Figure 8-23) then an electronic 7131 Request cannot be submitted.

Note: Notice of Discharge, Hospital Summary, Certificate (21-Day), and Admission Report are NOT available if the user selects Outpatient/Activity. These options are only available if Admission is selected.

The screenshot shows the 'New 7131 Request' form. The 'Type of request:' section has two radio buttons: 'Outpatient/Activity' (selected and circled in red) and 'Admission'. Below this, there are several checkboxes for report types: 'Notice of Discharge', 'Hospital Summary', 'Certificate (21 Day)', 'Other/Exam (Review/Remarks)', 'Special Report', 'Competency Report', 'VA Form 21-2680', 'Asset Information', 'Admission Report', and 'Beginning Date Care'. The 'Request Date' is set to 'TODAY', 'Date Last Status Changed' is 'TODAY', and 'RO' is 'ST. PETERSBURG-RO'. At the bottom, there are buttons for 'Send 7131 Request' and 'Cancel Request'.

Figure 8-22

The screenshot shows the 'New 7131 Request' form. The 'Type of request:' section has two radio buttons: 'Outpatient/Activity' and 'Admission' (selected and circled in red). To the right, a 'Select Admission Date:' dropdown menu is open, showing a list of dates and times: '2006@03C/SURG', '2004@4A/TELEMETRY', '2004@NH-5CT', '2004@4A/TELEMETRY', '2004@4A/TELEMETRY', and '2004@4A/TELEMETRY'. The rest of the form, including the checkboxes for report types and the bottom buttons, is identical to Figure 8-22.

Figure 8-23

Step 4 The user checks the items to request i.e. **Pt. Name, Requested By, Request Date, Date Last Status Changed, RO, and Admission Date.**

Step 5 The user enters detail information concerning the request in the **Comments** area ([Figure 8-25](#)). A Release of Information Clerk at VHA must search old records for the user's request and appreciates any help the user can give by being specific about the request.

Step 6 The user clicks **Send 7131 Request** and the request is transmitted. Or, the user can click **Cancel Request** ([Figure 8-5](#)). CAPRI opens a dialog box asking the user to confirm the choice.

Step 7 To inquire about the status of a 7131 request or to generate a report for proof of a request to be placed in the claims file, the user selects the **Status Inquiry** option shown in the next section in 21.

Figure 8-25

8.3.3. 7131 Request Status Inquiry

Step 1 – The user logs into CAPRI, selects a patient, and clicks the **7131 Request** tab. The **7131 Requests:** screen displays any pending or completed 7131 requests ([Figure 8-6](#)).

Step 2 – The user selects the 7131 request which they would like to run the inquiry and clicks the **Status Inquiry** button.

Figure 8-26

Step 3 – The following report opens under the **Reports** tab (Figure 8-7). To check the status of another 7131 request, the user must go back to the **7131 Request** tab and start from **Step 2** of this section.

Step 4 – The user can print the report for the claim folder by choosing **Print** from **File** on the main menu.

The screenshot shows a window titled 'Reports' with tabs for 'Admin', 'Health Summaries', and 'Clinical Documents'. The 'Reports' tab is active. The window displays patient information: Patient Name (redacted), SSN (redacted), Activity Date: JUN 28, 2002@08:00, and Claim Number: SS. Below this is a table with columns: Requisition, Status, Status Date, Operator, and Current Division. The table contains one row with the following data: Requisition: Notice/Discharge: Hospital Summary: 21-day Certificate: Other/Exam: APR 17, 2003, Status: PENDING, Status Date: (blank), Operator: 1, and Current Division: (blank). Below the table, there is a section for 'REMARKS' which states: 'This is a sample 7131 request' and 'Requesting location: ST. PETERSBURG-RO'. The 'Date of Request:' field is also present but empty.

Figure 8-27

CAPRI Reports (Patient-Specific)

The **Reports** tab allows the user to make various patient inquiries, review registration and profile data, and request surgery reports for the selected veteran.

8.3.4. Patient Inquiry

This report provides demographic, eligibility, and treatment information about the selected patient.

8.3.5. Detailed Inpatient Inquiry

This report lists all of the patient's inpatient admissions, with the date, time, and ward. The user can select an admission and click **OK** to get detailed information about that admission. This includes admission and discharge dates, transfers between wards, care providers, and diagnosis.

8.3.6. C&P Exam Detail

This report was moved to the **C&P Exams** tab. If attempting to use this report, the user is directed to that tab.

8.3.7. 7131 Detail

This report was moved to the **7131 Request** tab. If attempting to use this report, the user is directed to that tab.

8.3.8. Additional Treating Facilities

This report shows if the patient was treated at any VHA facilities other than the one currently connected to by the user.

8.3.9. View Registration Data

This report provides full demographic data, including military information, for the selected patient.

8.3.10. Patient Profile Medical Administration Service (MAS) (Full)

The following screen is the default – it specifies all dates, appointments, enrollments, team information, edits, dispositions, and the means test. The user can change the generated report to exclude particular types of information by selecting **No** for that particular type.

8.3.11. Surgery Report

This option generates a list of all available surgery reports for the selected patient. If the user selects a procedure from the list, the **OK** button is enabled.

8.3.12. Other Patient-Specific Reports

As noted below in the non-patient-specific reports section, if a veteran is selected prior to choosing Reports from the File menu, two of the reports have additional options. The **Reprint a 21-Day Certificate** and the **Reprint a Notice of Discharge** reports allow the user to choose either for a date or for the patient selected. Refer to the next section for details on the other reports accessed from the **Reports** option under the **File** menu.

8.3.13. Reprint a 21-Day Certificate

This option is used to reprint a 21-Day Certificate for a particular patient. After selecting a patient, the user accesses Reports under the File menu. The resulting report dialog box includes that patient as a selection option. If the user does not have a patient selected, the dialog box does not include this as an option. The certificate produced is exactly the same as the original certificate. ROC 119, which appears at the bottom of each certificate, stands for VA Form 119 Report of Contact.

8.3.14. Reprint a Notice of Discharge

It may be occasionally necessary to reprint a Notice of Discharge for a patient. After selecting a patient, the user accesses **Reports** under the **File** menu. The resulting report dialog box includes that patient as a selection option. If the user did not have a patient selected, the dialog box would not include this as an option. If the admission associated with the 7131 was deleted and notification already sent, a message is displayed. The message includes the patient's name, SSN, date and time of admission, notice that the admission has been deleted, and a recommendation to contact the medical center.

CAPRI Reports (Non-Patient-Specific)

The Non-Patient-Specific CAPRI Reports are accessed by selecting **Reports** under **File** on the main menu.

8.3.15. C&P Exams Reports

8.3.15.1. AMIS 290

This option electronically produces the AMIS C&P report that was manually produced by the regional office. It is a general system of computer programs used to process management reports. The AMIS 290 report covers compensation and pension examination request activity. The regional office AMIS 290 calculates the data based only on that specific regional office's requests, or when the regional office is left blank, it calculates the data for all regional offices. The average processing time reported on the AMIS 290 report now accounts for lost 2507 request processing time due to appointment reschedules at the request of the veteran. Processing time for an insufficient request includes the processing time of the original request. In addition to a hard copy being produced, this option allows the user to send a MailMan message either locally or via network mail. The mail bulletin contains the same information that appears on the report.

By selecting one of the following filters to run the AMIS 290 report, CAPRI produces separate statistics as indicated:

- Agent Orange
- Benefits Delivery at Discharge and Quick Start
- IDES
- All priorities except AO, BDD, IDES , and QS.

| **Figure 8-28** shows the selection appropriate to produce an AMIS 290 report for all priorities except AO, BDD, IDES and QS. As appropriate, this report would include statistics for **Priority of Exam(s)** types:

- TERMINAL
- POW
- ORIGINAL SC
- ORIGINAL NSC
- INCREASE
- REVIEW
- OTHER
- INSUFFICIENT EXAM

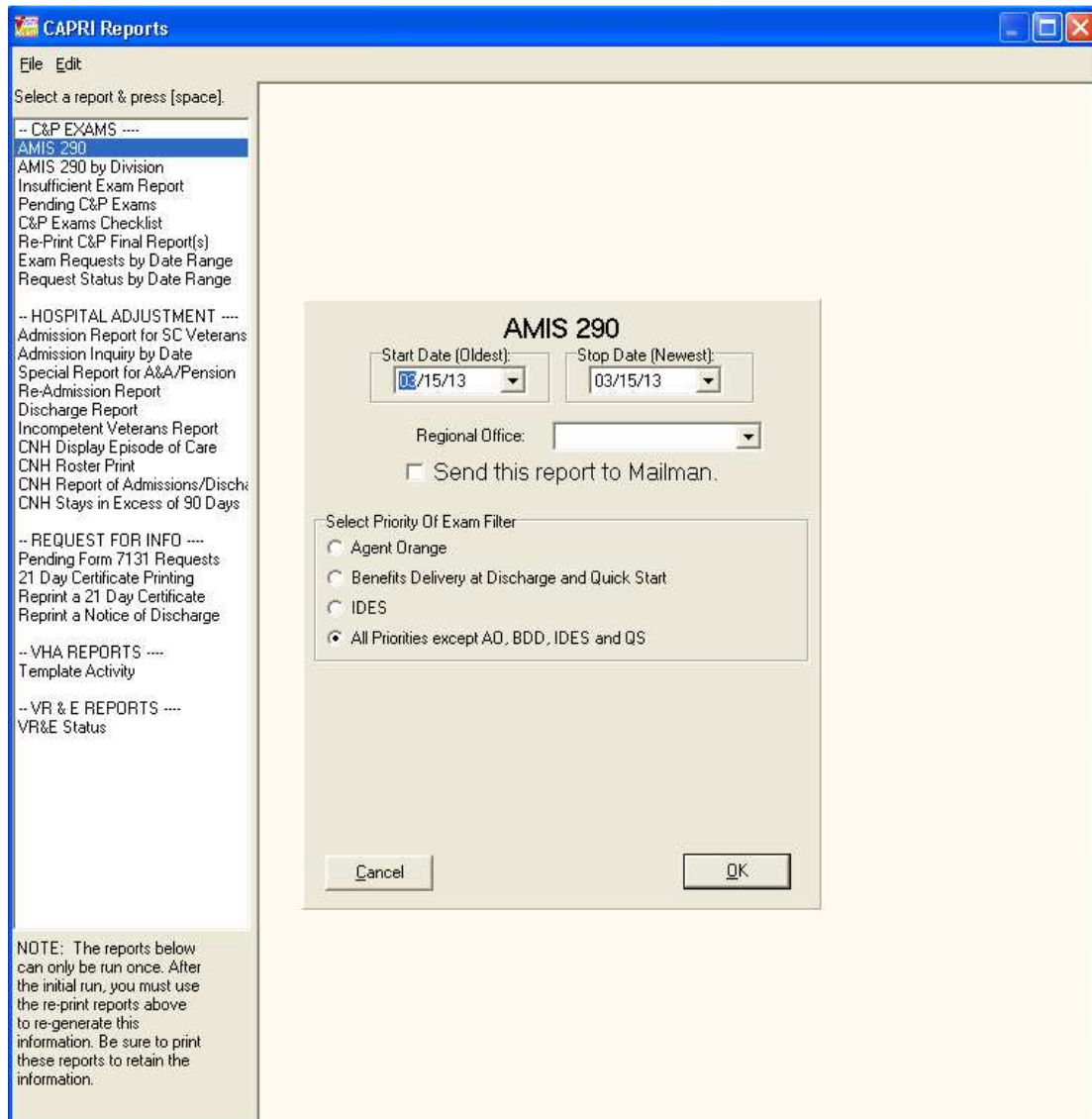


Figure 8-28

Figure 8-29 shows the output from the AMIS 290 report.

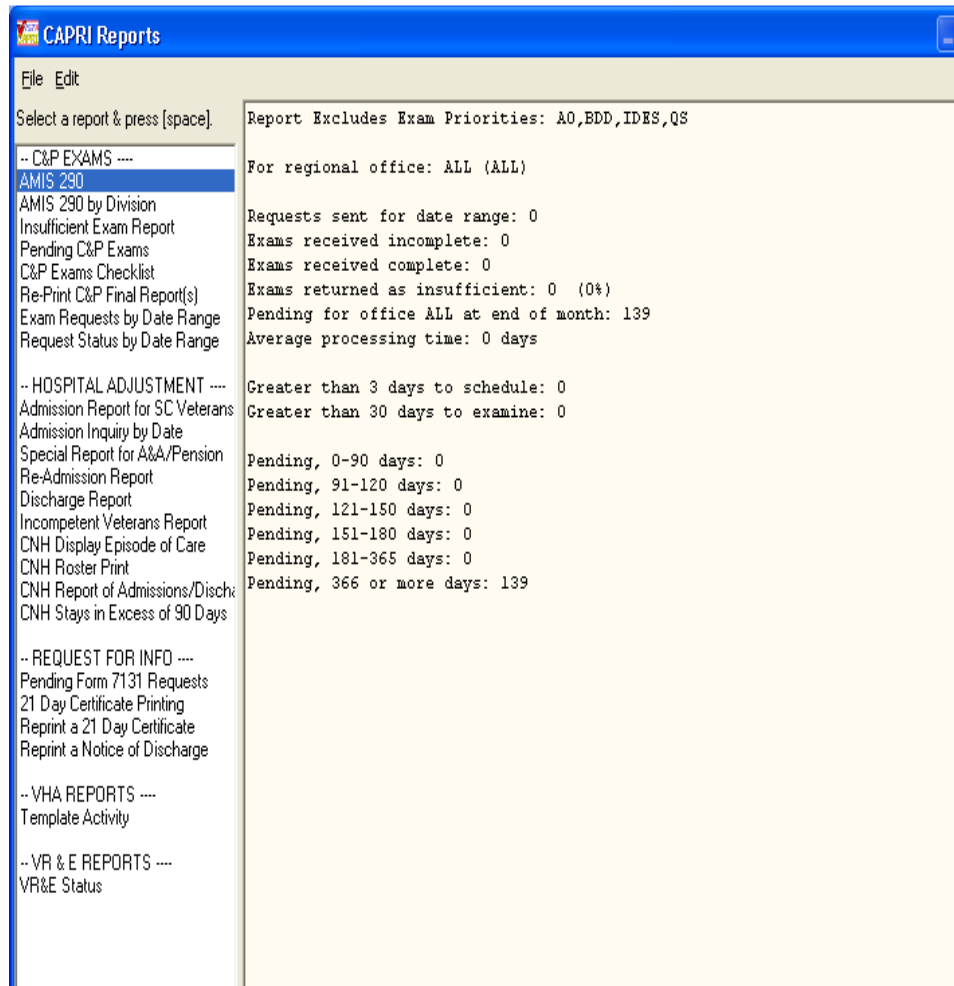


Figure 8-29

The terminology below may help interpret the AMIS 290 and Insufficient Exam Report.

1. **Total Pending from Previous Month**

This is the total number of 2507 requests reported to MAS before the last day of the previous month that were not completed prior to the end of that month, or were completed prior to the end of that month and reopened.

2. **Requests Received for Date Range**

Total number of requests that were not entered with a priority of exam equal to INSUFFICIENT and had a date reported to MAS that fell within the date range entered by the user.

3. **Exams Returned as Insufficient**

Total number of requests that were entered with a priority of exam equal to INSUFFICIENT and had a date reported to MAS that fell within the date range entered by the user.

4. **Requests Returned Complete**

Total number of requests that have a date released that fell within the date range entered and have a request status of COMPLETED, PRINTED BY RO or RELEASED TO RO, NOT PRINTED.

5. **Requests Returned Incomplete**

Total number of requests that have a cancellation date that falls within the date range entered and a request status that equals CANCELLED BY RO or CANCELLED BY MAS.

8.3.15.2. AMIS 290 by Division

This is the same report as the AMIS 290 above, but the user can limit the report to a single division of the VHA facility that performs C&P examinations, such as a community-based clinic.

8.3.15.3. Insufficient Exam Report

The **Insufficient Exam Report** option prints a report of 2507 requests entered with a priority of INSUFFICIENT EXAM for a specified date range. The user may choose a detailed or summary version of the report. Only exam reasons and types that have information to report are included on the detailed version of the report.

The summary version of the report is divided into two parts. The first part contains the total number of 2507 requests/exams received for the date range, the total number of priority insufficient requests/exams for the date range, and the percentage of insufficient requests/exams received. Due to the rounding of the component percentages, the total of the percentages may not equal 100%. The second part of the summary version is a breakdown of each reason an exam was returned. The detailed version allows the user to display one, many, or all insufficient reasons and AMIE exams.

Other information provided includes exam type, patient name, SSN, and claim number. Provider and exam date on this report is the provider and date from the originally completed 2507. The exam date is not included if the original 2507 has been purged. The length of the veteran's name and the provider are limited to 15 characters each. If either field has been truncated, it appears with two asterisks (**). If an insufficient 2507 is transferred from one site to another, that exam is reported on the insufficient exam report for both sites (original and remote).

Please Note** If Summary is chosen, CSV Comma Delimited Export will “not” be available.

8.3.15.4. Pending C&P Exams

This option prints out all pending C&P requests. The user may sort the reports by request status, routing location, veteran name, or age of the request.

When sorting by status, the report can be tailored to identify NEW, PENDING, TRANSCRIBED, or ALL pending C&P exams. Sorting by TRANSCRIBED may be useful for identifying ready to rate claims that have been completed but not released to the RO. Common delays for not releasing transcribed results include waiting for a physician's signature and/or waiting for additional required tests or studies. An inquiry to the VAMC C&P clinic can clarify reasons as to why TRANSCRIBED results have not been released if an excessive number of days have elapsed since the date of the C&P exam request.

Each report displays the following information, if applicable: veteran name, SSN, claim number, request date, elapsed days, exams requested, and requester name and location. The total number of exams pending is also provided.

8.3.15.5. C&P Exams Checklist

The **Exam Check List for RO** option is used to print a checklist used by regional office personnel to select compensation and pension examinations for veterans. The request worksheet lists the body systems and the exam worksheet names. It also contains a section for remarks. The top portion of the work sheet allows the requester to enter veteran-specific information including:

- Veteran's name, SSN, and Claim Number (C-Number)
- VAMC where the exam is to be performed
- Veteran's day and night telephone numbers
- Power of Attorney
- Date the exam was ordered and by whom
- Insufficient exam date

8.3.15.6. Re-Print C&P Final Report(s)

This option allows the reprinting of final 2507 exams with the status of **Completed, printed by RO**. The reports are sorted by the last two digits of the claim number. The user must enter the date the report was previously printed. Reprinting a request is not allowed unless the person requesting the reprint has a division which matches the station number of the requesting regional office. The exam must have the status **Completed, printed by RO** or **Released to RO, not printed**. The package is designed to print any lab/radiology results designated for C&P. When printing, the system examines all lab/radiology results for 120 days before the release date. The output includes a summary portion that includes patient name, SSN, claim number, and request date. The total number of requests to be printed is also provided.

8.3.15.7. Exam Requests by Date Range

This report generates a simple list of all exam requests entered within the specified date range. The report is sorted by date of entry. The following fields are reported: **SSN, Patient Name, Request Date, Date Released, Date Printed**, and **Status**. The report was created primarily to assist VHA in tracking exams that may have been requested but not released.

C&P exam Status comprises the following exam request categories:

- **New** exam requests from VBA
- **Pending, Reported** exam requests acknowledged by VHA
- Canceled by MAS
- Canceled by RO
- Released to RO, Not Printed
- Completed, Printed by RO

8.3.15.8. Request Status by Date Range

The report in **Error! Reference source not found.** generates a simple list of exam requests by Request Status entered within the specified date range. In addition, the statuses of **New**;

Pending, Reported; and Released to RO, Not Printed may be run for a blank date range by choosing **Yes** to **Blank Date Range**. The following fields are reported: **SSN, Patient Name, Request Date, Date Released, Date Printed, Status, Date Cancelled, and VBA Requesting Station**.

The report is sorted by the date of the last request status change. When a blank date range is used for **New; Pending, Reported; or Released to RO** the report is sorted by request date. The report was created primarily to assist VBA to better track exam requests as they are processed.

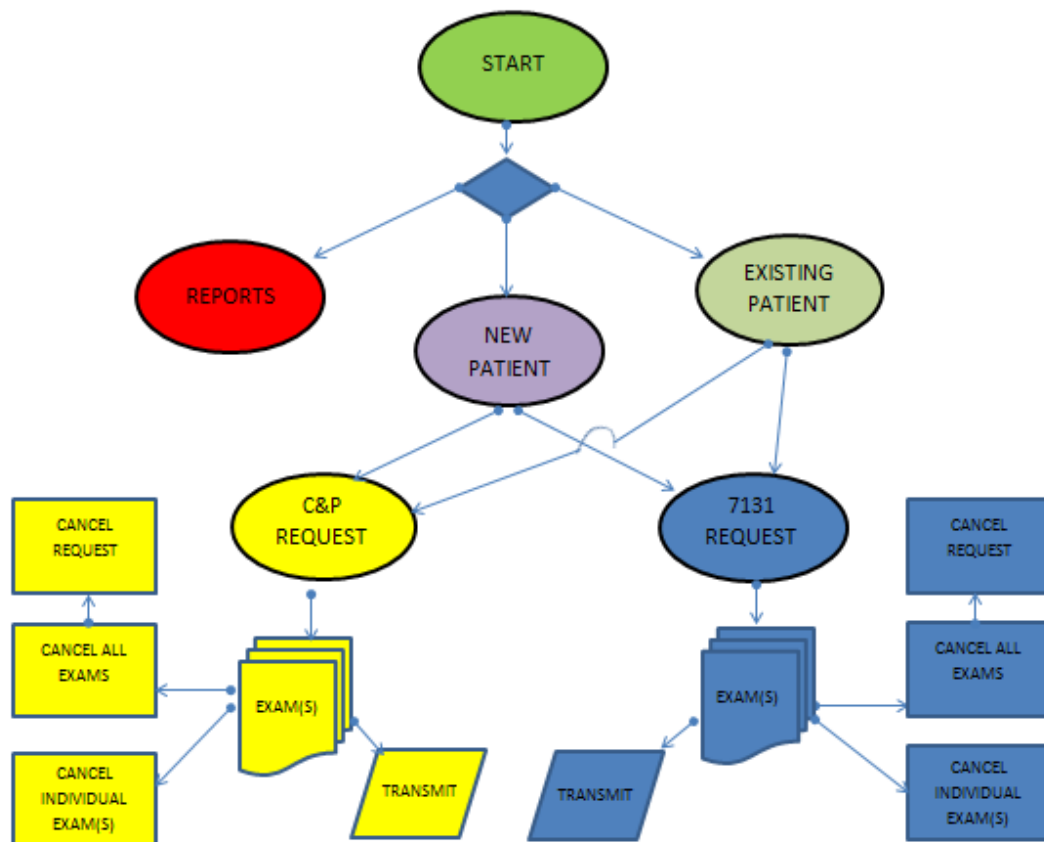
Request Status by Date Range allows the following status filters:

- All Statuses
- New
- Pending, Reported
- Canceled by MAS
- Canceled by RO
- Released to RO, Not Printed
- Completed, Printed by RO

The report can also be exported to Excel in a Comma Separated Value (.CSV) file.

8.4. Navigation Hierarchy

Provide a diagram of the navigation hierarchy that shows how a user moves through the GUI.



8.4.1. Start- Capri Select a Patient

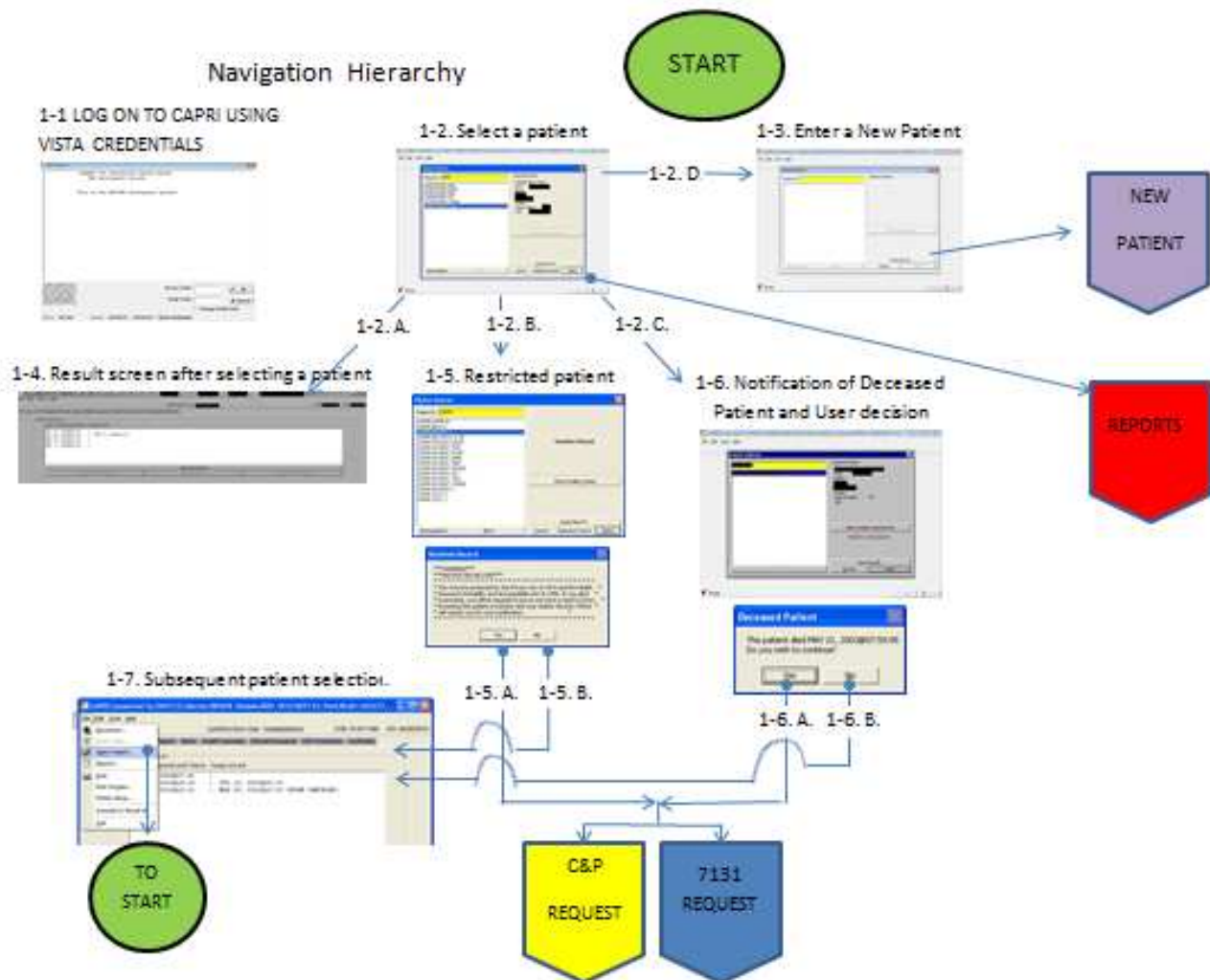


Figure 8-30

8.4.2. Start-Capri Create a New Patient

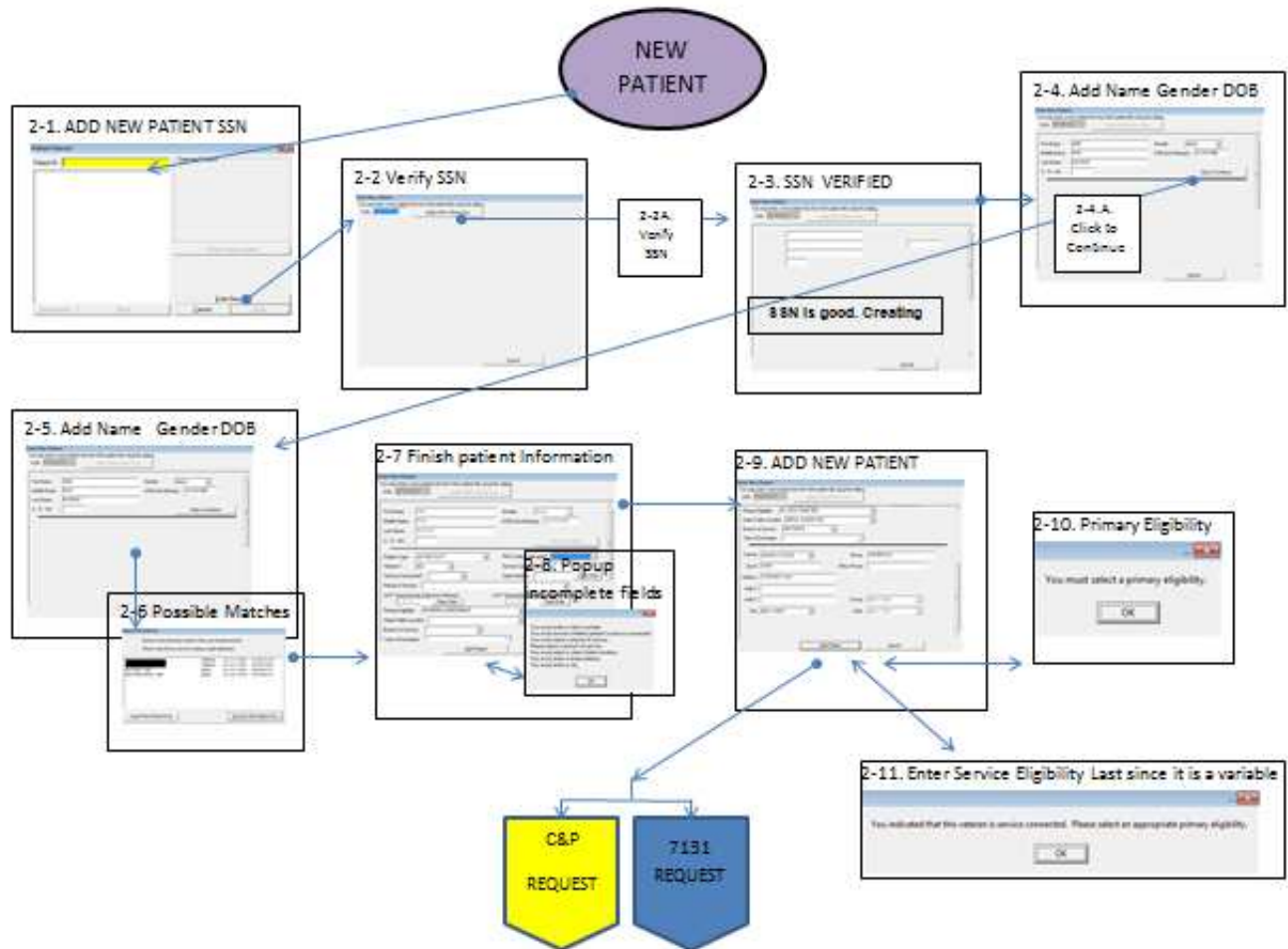
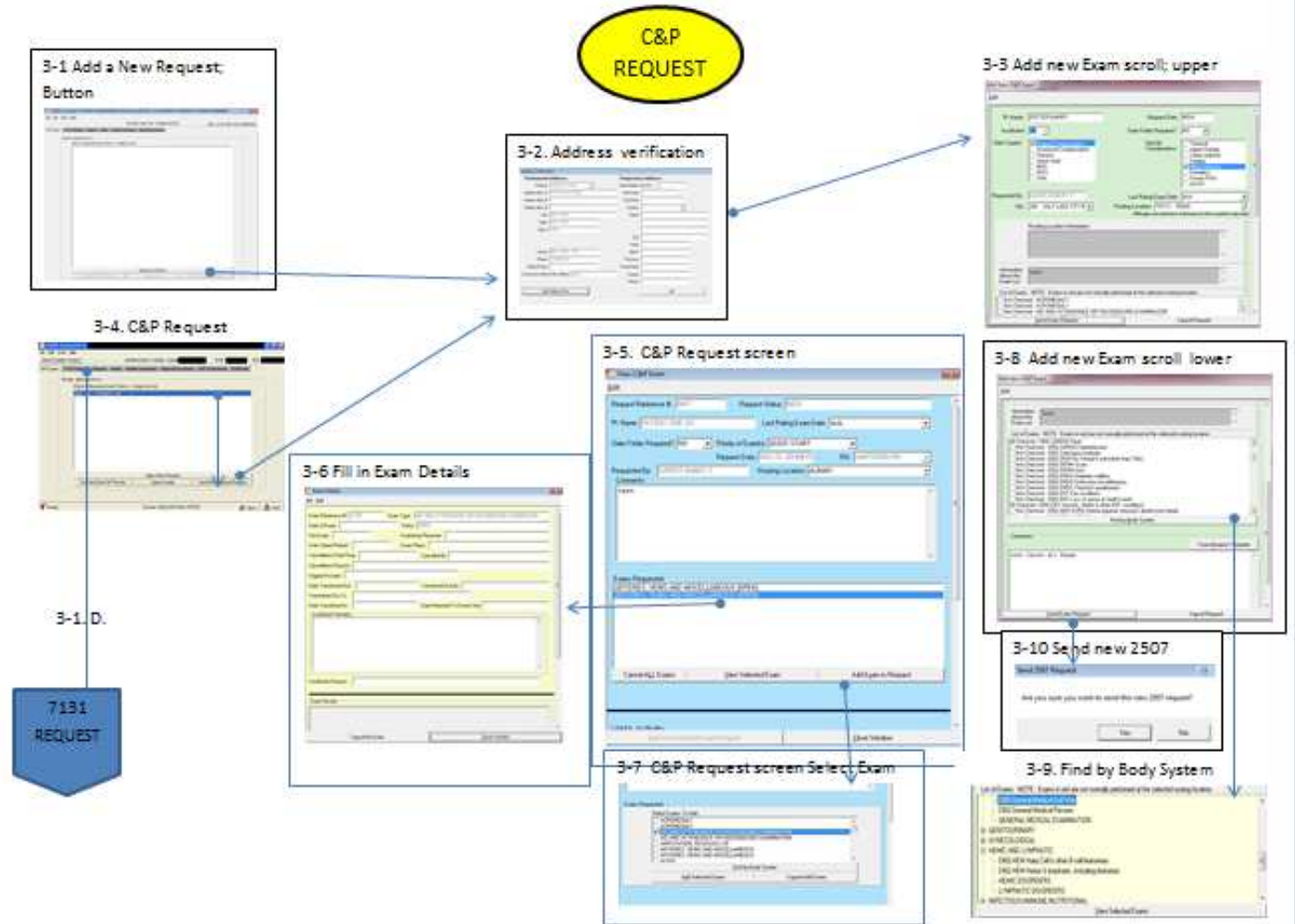


Figure 8-31

CAPRI System Design Document



CAPRI System Design Document

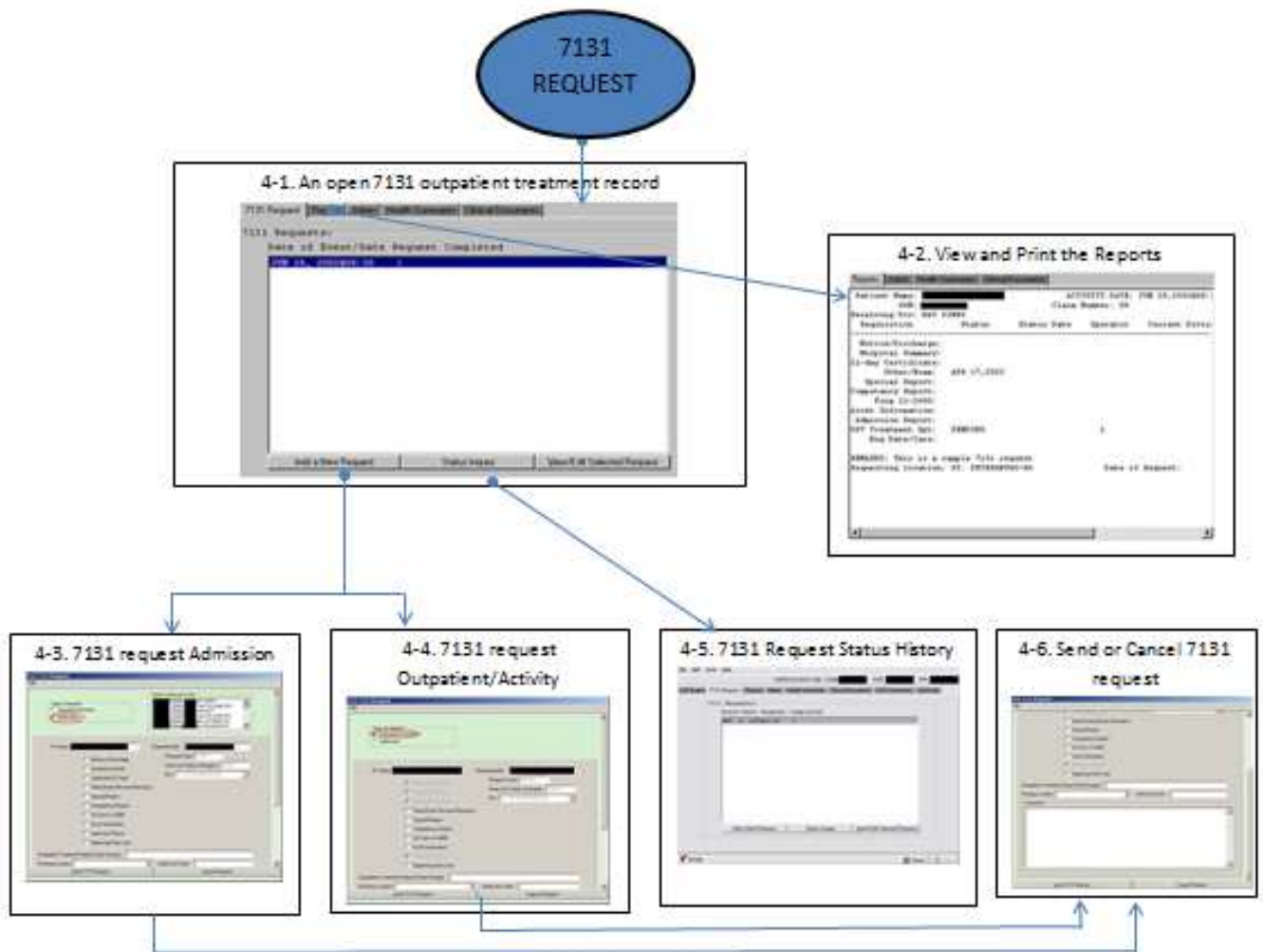


Figure 8-33

8.4.5. Reports

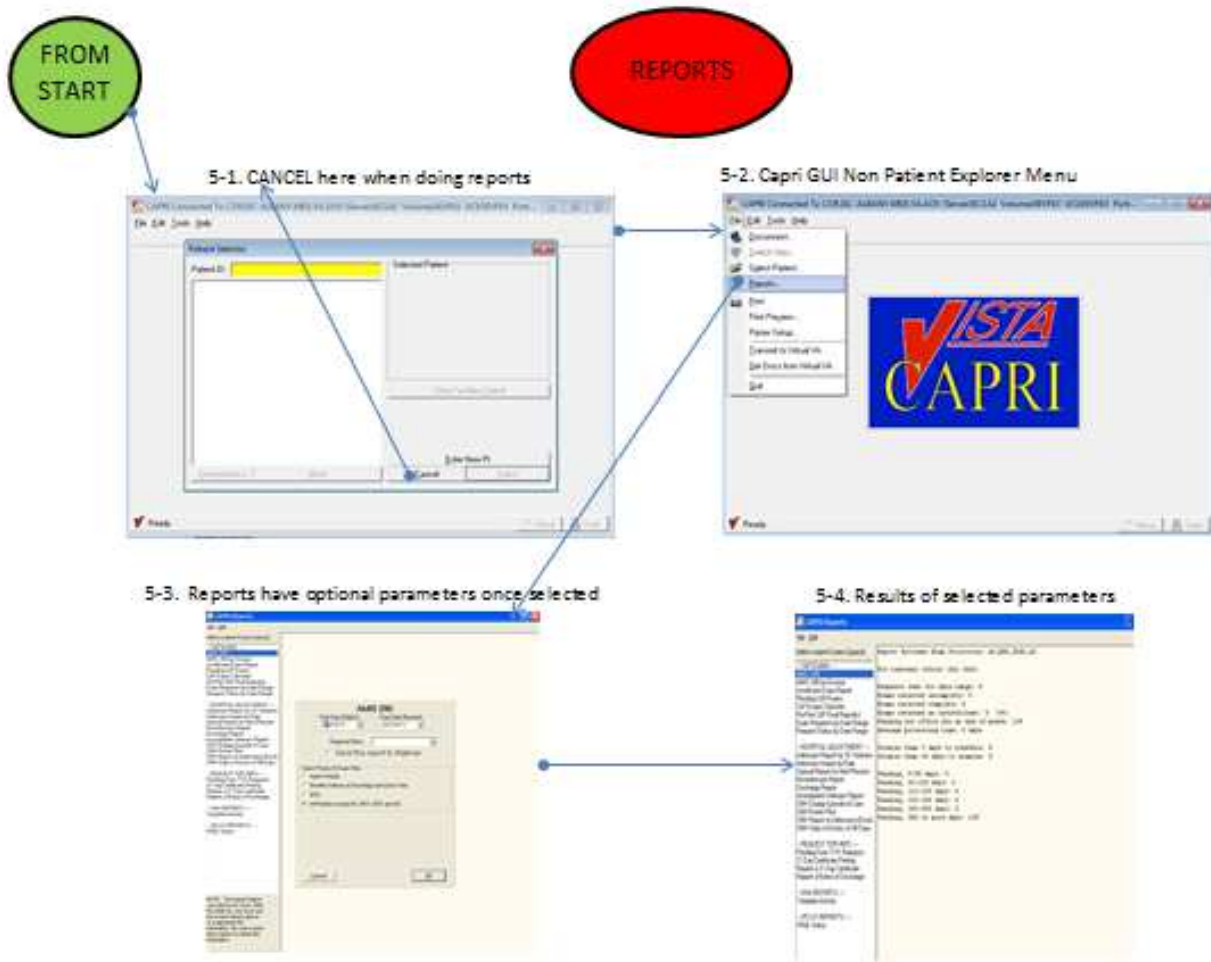


Figure 8-34

8.4.6. Cancellations

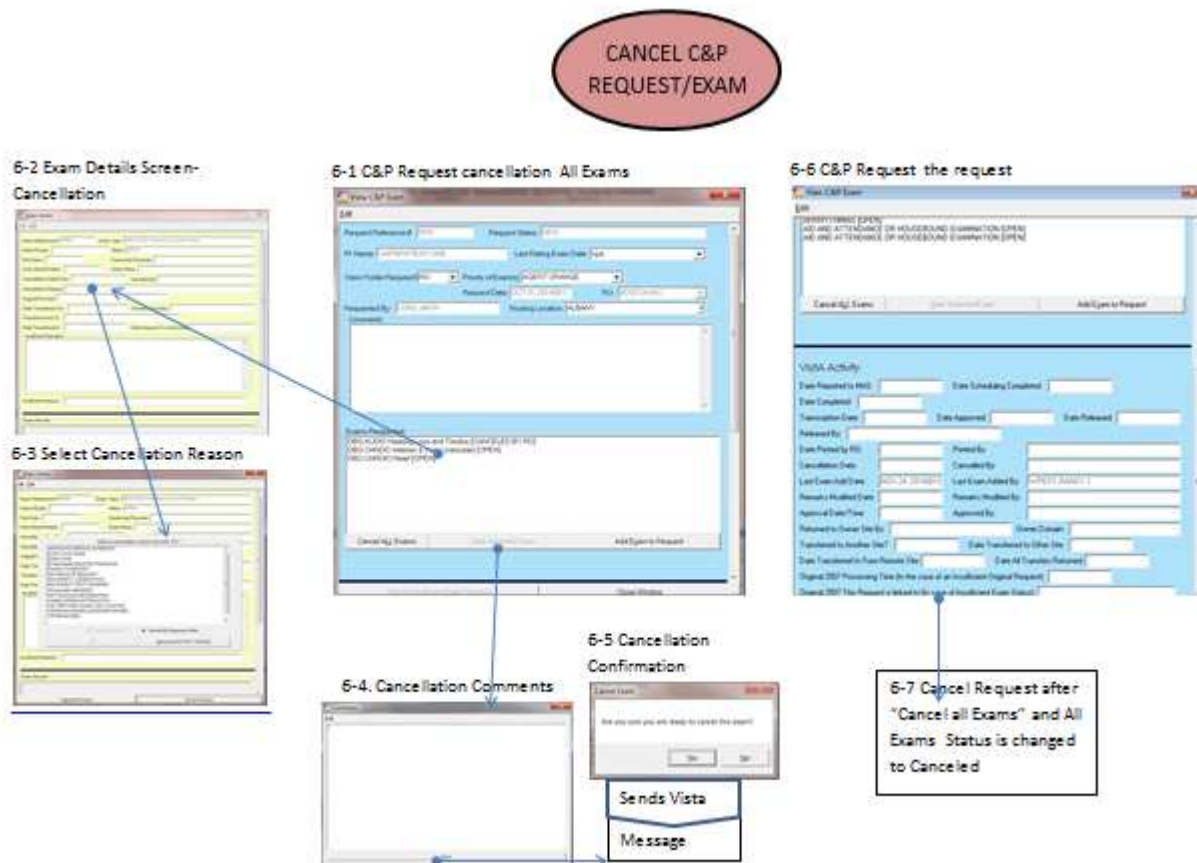


Figure 8-35

Table 8-2

ID	Name	Description	Interface Name	Interface System
1-1	LOG ON	Log on to the Capri GUI	CAPRI	VISTA
1-2	Select a Patient	Enter a SSN or Name, If a list is presented this may be an existing pateint		
1-2.A	Path	Possible Result	N/A	N/A
1-2.B	Path	Possible Result	N/A	N/A
1-2.C	Path	Possible Path	N/A	N/A
1-2.D	Path	Possible Path	N/A	N/A
1-3	Enter a New Patient	If the patient is not in the system it will follow a New Patient Entry Path	C&P Pateint Selector	CAPRI GUI Form

ID	Name	Description	Interface Name	Interface System
1-4	Result screen after selecting a patient	List of Exams for the selected patient	Search Results Tabs	CAPRI GUI
1-5	Restricted Patient	The patients records may be restricted and the user is notified and given the option to procede.	Restricted Pateint Popup	CAPRI GUI
1-6	Notification of Deceased Patient and User decision	The patient may be deceased and the user is notified and given the option to procede	Deceased Pateint Popup Message	CAPRI GUI
1-5.A	Path	Possible Path to procede with restricted patient. Selecting to procede with Restricted allows for selection of Requests	N/A	N/A
1-5.B	Path	Possible Path to NOT procede with restricted patient. QUITs	N/A	N/A
1-6.A	Path	Possible Path to procede with Deceased patient record. Selecting to procede with Restricted allows for selection of Requests	N/A	N/A
1-6.B	Path	Possible Path to NOT procede with Deceased patient record	N/A	N/A
1-7	Subsequent patient selection	Selecting to procede brings the user back here to select a patient	Pateint Selector Form	CAPRI GUI
2-1	ADD NEW PATIENT SSN	From the Figure 8-31, When the SSN is not already entered in the VISTA systems, the path to enter a new patient begins	Pateint Selector Form	CAPRI GUI
2-2	Verify SSN	When an SSN is entered the Verify SSN is not in Use button is available. CAPRI functions will check through Fileman Calls or Remote Procedure to see if the patient is already in the system.	SSN Verification Search Function	CAPRI GUI
2-3	SSN Verified	When the SSN is verified as being Unique; the user is notified with a popup message "SSN is good. Creating"	CAPRI Popup Message	CAPRI GUI
2-4	Add Name Gender DOB	Continue creating the patient entry by entering the basic Demographics	Enter NEW Patient	CAPRI GUI
2-5	Click to Continue	After entering the new data, "Click to continue" and CAPRI functions will check through Fileman Calls or Remote Procedure to see if the patient is already in the system again	Enter NEW Patient "Click to Continue" Button	CAPRI GUI
2-6	Possible Matches	After the search, a list of possible matches is presented to help avoid duplicate entries.	CAPRI POP up FORM	CAPRI GUI

ID	Name	Description	Interface Name	Interface System
2-7	Finish Patient Information	After verified the status of the form changes to allow additional Demographics panels.	Enter New Patient Secondary Status WorkFlow view	CAPRI GUI
2-8	Popup Incomplete fields	The form will not save if fields are incompleated. The Popup will present a list of missing fields until all are completed	CAPRI POP up FORM	CAPRI GUI
2-9	ADD NEW PATIENT	After all necessary fields are populated and the ADD Patient button is pressed	Enter New Patient Form	CAPRI GUI
2-10	Primary Eligibility	This will popup if the primary eligibility is not selected or incorrect	CAPRI POPUP	CAPRI GUI
2-11	Enter Service Eligibility	This should be entered last. If it is selected to early and later and incompatible Primary eligibility is entered the patient will not be Added. Even though the fields are populated The exact reason is not presented just this message when ADD PATIENT button is pressed.	CAPRI POPUP message	CAPRI GUI
3-1	Add a New Request Button	C&P Exam Add a new Request Tab, Select the Add a new request button	CAPRI Selection Tab form	CAPRI GUI
3-2	Address Verification	Before a New Exam is entered or an existing exam is selected the address verification screen form is presented to ensure the VA has the most current address on file	CAPRI Address Verification Form	CAPRI GUI
3-3	Add a new Exam Scroll: Upper	This form is long so scrolling is required to see the full exam request. This is the upper section	Add a New C&P Exam "Edit" form	CAPRI GUI
3-4	C&P Request	There are optional Paths from this screen depending on the Tab selected. It was placed here to demonstrate another place the Address Verification form is prompted.	C&P Request	CAPRI GUI
3-5	C&P Request Screen	Displayed after address verification, to edit or enter a request	Add a New C&P Exam "Edit" form	CAPRI GUI
3-6	Fill in Exam Details	Select an exam from the C&P Request. Double click the exam name in the Exam Request window and the Exam Details Form is presented	Add a New C&P Exam "Edit Details" form	CAPRI GUI

ID	Name	Description	Interface Name	Interface System
3-7	C&P Request screen Select Exam	When the user selects the “Add Exam to Request” button. A list of available exams list (in alphabetical order) is presented for selection. This takes the place of the previously entered exams list that was in this field.		CAPRI GUI
3-8	Add a new Exam Scroll: Lower	This form is long so scrolling is required to see the full exam request. This is the lower section.	Add a New C&P Exam “Edit” form	CAPRI GUI
3-9	Find By Body System	This list is a way to narrow the search for an exam title by the Body Systems category. Select the “Find by Body System” button to bring up a list of Body Systems	Add a New C&P Exam “Edit” form option	CAPRI GUI
3-10	Send Exam Request	This verification popup is presented after the “Send Exam Request” is selected.	Add a New C&P Exam “Edit” form Pop UP	CAPRI GUI
4-1	An Open Outpatient treatment record	Highlighted is the 7131 request tab	7131 Report Form	CAPRI GUI
4-2	View Print Reprts Tab	This is a Reports tab for the 7131 Request	Reports tab	CAPRI GUI
4-3	7131 Request Admission	Adding a 7131 request for an “Admission” has an added field for selecting admission date	New 7131 Request Form	CAPRI GUI
4-4	7131 Request Outpatient/Activity	Adding a 7131 request for an “Outpatient/Activity” has an no admission date field and notice of discharge, Hospital Summary;Certificate 21 Day; and Admissoin report are not available for selection	New 7131 Request Form	CAPRI GUI
4-5	7131 Request Summary	The 7131 tab will contain a list of events and date completed.	7131 Request Tab	CAPRI GUI
4-6	Send or Cancel Request	The lower portion of the 7131request form is where comments are entered and the request is sent. Or cancelled.	New 7131 Request Form	CAPRI GUI
5-1	Cancel here when doing reports	After logging in; the Patient Selector is automatically presented. Cancel this form to select from the File menu.	Capri Pateint Selector	CAPRI GUI
5-2	Capri GUI Non Patient Explorer Menu	Select Reports from the File Menu	CAPRI GUI File menu	CAPRI GUI
5-3	Reports have optional parameters once selected	Reports have optional parameters set by the user depending on the report selected	CAPRI GUI Reports Tab	CAPRI GUI

ID	Name	Description	Interface Name	Interface System
5-4	Results of selected parameters	One example for a selected report	CAPRI GUI Reports Tab Results Form	CAPRI GUI
6-1	C&P Request cancellation All Exams	Cancellation of a C&P request can either be individual exams of cancel all exams at once	View C&P Exams form	CAPRI GUI
6-2	Exam Details Screen-Cancellation	Here one exam has been selected. To cancel it enter a cancellation date	Exam details form	CAPRI GUI
6-3	Select Cancellation Reason	After a Cancellation Date has been entered the user is required to select a cancellation reason	Cancellation Reasons Popup Drop down list Form	CAPRI GUI
6-4	Cancellation Comments	Optional cancellation comments can be entered here	Cancellation Comments Popup	CAPRI GUI
6-5	Cancellation Confirmation	Confirming your intention to cancel the exam will cancel the exam and send a vista message containing the comments if any are made.	Cancellation Confirmation Popup	CAPRI GUI
6-6	C&P Request Cancel the request	Cancellation of the entire request occurs when the cancel all exams is selected and the cancellation date is entered in the request form	C&P Request Cancel the request	CAPRI GUI
6-7	Note: After all exams status' are changed to Canceled you can Cancel the Request	This note is a reminder that the request cannot be cancelled until each exam's status is changed to cancelled in the status field	View C&P exams	CAPRI GUI

9. Security and Privacy

9.1. Security

ID	FAMILY	ID	FAMILY
AC	Access Control	MP	Media Protection
AT	Awareness and Training	PE	Physical and Environmental Protection
AU	Audit and Accountability	PL	Planning
CA	Security Assessment and Authorization	PS	Personnel Security
CM	Configuration Management	RA	Risk Assessment
CP	Contingency Planning	SA	System and Services Acquisition
IA	Identification and Authentication	SC	System and Communications Protection
IR	Incident Response	SI	System and Information Integrity
MA	Maintenance	PM	Program Management

The Above table is from [NIST 800-53 revision 4](#) reference this publication for detailed descriptions of the following Security Controls.

9.1.1. Security Control Families

CAPRI adheres to the VA Information Technology System Policies. These policies address the following Security Control attributes.

9.1.1.1. Access Control

Access Control Policy and Procedures; Account Management; Separation of Duties; Least Privilege; Unsuccessful Logon Attempts; System Use Notification; Previous Logon (Access) Notification indicator determines access recertification requirements; Concurrent Session Control limits the amount of concurrent sessions allowed by individuals; Session Lock limits the concurrent access of individual records. Session Termination; Permitted Actions without Identification or Authentication; Security Attributes; Remote Access; Wireless Access; Access Control for Mobile Devices; Use of External Information Systems; Information Sharing; Publicly Accessible Content; Data Mining Protection provided by VA Network Security; Access Control Decisions are made by the Information Security Office; Reference Monitor.

9.1.1.2. Awareness and Training

All personnel receive annual training on Security Awareness and Training Policy and Procedures, Security Awareness Training. Additional role based training is provided by the users service. Security training records are maintained on the training management system.

9.1.1.3. Audit and Accountability

Audit and Accountability Policy and Procedures; Audit Events; Content of Audit Records; Audit Storage Capacity; Response to Audit ; Processing Failures; Audit Review, Analysis, and Reporting; Audit Reduction and Report Generation; Time Stamps; Protection of Audit Information; Non repudiation; Audit Record Retention; Audit Generation; Monitoring for Information Disclosure; Session Audit; Alternate Audit Capability; Cross Organizational Auditing

The following instructions are to view your audit log. This is an example only, use your correct information to locate your log.

Example: The log file is named DVBA_2.7_BuildVersion_dd_mm_yy.TXT

Go to: C:\Documents and Settings\YourVAUserName\Local Settings\Temp\DVBA_2.7_BuildVersion_dd_mm_yy.txt

9.1.1.4. Security Assessment and Authorization

Security Assessment and Authorization Policies and Procedures; Security Assessments; System Interconnections; Plan of Action and Milestones; Security Authorization; Continuous Monitoring; Penetration Testing; Internal System Connections

9.1.1.5. Configuration Management

Configuration Management Policy and Procedures; Baseline Configuration; Configuration Change Control; Security Impact Analysis; Access Restrictions for Change; Special Publication 800 53 Revision 4 Security and Privacy Controls for Federal Information Systems and Organizations; Configuration Settings; Least Functionality; Information System Component Inventory; Configuration Management Plan; Software Usage Restrictions; User Installed Software

9.1.1.6. Contingency Planning

Contingency Planning Policy and Procedures; Contingency Plan; Contingency Training; Contingency Plan Testing; Alternate Storage Site; Alternate Processing Site; Telecommunications Services; Information System Backup; Information System Recovery and Reconstitution; Alternate Communications Protocols; Safe Mode; Alternative Security Mechanisms

9.1.1.7. Identification and Authentication

Identification and Authentication Policy and Procedures; Identification and Authentication (Organizational Users); Device ; Identification and Authentication; Identifier Management; Authenticator Management; Authenticator Feedback; Cryptographic Module Authentication; Identification and Authentication (Non Organizational Users); Service Identification and Authentication; Adaptive Identification and Authentication; Re authentication

9.1.1.8. Incident Responses

Incident Response Policy and Procedures; Incident Response Training; Special Publication 800 53 Revision 4 Security and Privacy Controls for Federal Information Systems and Organizations; Incident Response Testing; Incident Handling Incident Monitoring; Incident Reporting; Incident Response Assistance; Incident Response Plan; Information Spillage Response; Integrated Information Security Analysis Team

9.1.1.9. Maintenance

System Maintenance Policy and Procedures; Controlled Maintenance; Maintenance Tools; Nonlocal Maintenance; Maintenance Personnel; Timely Maintenance

9.1.1.10. Media Protection

Media Protection Policy and Procedures; Media Access; Media Marking/; Media Storage; Media Transport; Media Sanitization; Media Use; Media Downgrading

9.1.1.11. Physical and Environmental Protection

Physical and Environmental Protection Policy and Procedures; Physical Access Authorizations; Physical Access Control; Access Control for Transmission Medium; Access Control for Output Devices; Monitoring Physical Access; Visitor Access Records; Power Equipment and Cabling; Emergency Shutoff; Emergency Power; Emergency Lighting; Fire Protection; Temperature and Humidity Controls; Water Damage Protection; Delivery and Removal; Alternate Work Site; Location of Information System Components; Information Leakage; Asset Monitoring and Tracking;

9.1.1.12. Planning

Security Planning Policy and Procedures; System Security Plan; Rules of Behavior; Security Concept of Operations; Information ; Security Architecture; Central Management

9.1.1.13. Personnel Security

Personnel Security Policy and Procedures; Position Risk Designation; Personnel Screening; Personnel Termination; Personnel Transfer; Access Agreements; Third Party Personnel Security; Personnel Sanctions

9.1.1.14. Risk Assessment

Risk Assessment Policy and Procedures; Security Categorization; Risk Assessment; Vulnerability Scanning Technical Surveillance Countermeasures Survey

9.1.1.15. System and Services Acquisition

System and Services Acquisition Policy and Procedures; Allocation of Resources; System Development Life Cycle; Acquisition Process; Information System Documentation; Security Engineering Principles; External Information System Services; Developer Configuration Management; Developer Security Testing and Evaluation; Supply Chain Protection; Trustworthiness; Criticality Analysis; Development Process, Standards, and Tools; Developer Provided Training; Developer Security Architecture and Design; Tamper Resistance and Detection; Component Authenticity; Customized Development of Critical Components; Developer Screening; Unsupported System Components

9.1.1.16. System and Communications Protection

System and Communications Protection Policy and Procedures; Application Partitioning; Security Function Isolation; Information in Shared Resources; Denial of Service Protection; Resource Availability; Boundary Protection; Transmission Confidentiality and Integrity; Network Disconnect; Trusted Path; Cryptographic Key Establishment and Management; Cryptographic Protection; Collaborative Computing Devices; Transmission of Security Attributes; Public Key Infrastructure Certificates; Mobile Code; Voice Over Internet Protocol; Secure Name /Address Resolution Service(Authoritative Source); Secure Name /Address Resolution Service(Recursive or Caching Resolver); Architecture and Provisioning for Name/Address Resolution Service; Session Authenticity; Fail in Known State; Thin Nodes; Honeypots; Platform Independent Applications; Protection of Information at Rest; Heterogeneity; Concealment and Misdirection; Covert Channel Analysis; Information System Partitioning; Non Modifiable Executable Programs; Honeyclients; Distributed Processing and Storage; Out of Band Channels; Operations Security; Process Isolation; Wireless Link Protection; Port and I/O Device Access; Sensor Capability and Data; Usage Restrictions; Detonation Chambers

9.1.1.17. System Information Integrity

System and Information Integrity Policy and Procedures; Flaw Remediation; Malicious Code Protection; Information System Monitoring; Security Alerts, Advisories, and Directives; Security Function Verification; Software, Firmware, and Information Integrity; Spam Protection; Information Input Validation; Error Handling; Information Handling and Retention; Predictable Failure Prevention; Non Persistence; Information Output Filtering; Memory Protection; Fail Safe Procedures

9.1.1.18. Program Management

Information Security Program Plan; Senior Information Security Officer; Information Security Resources; Plan of Action and Milestones Process; Information System Inventory; Information Security Measures of Performance; Enterprise Architecture; Critical Infrastructure Plan; Risk Management Strategy; Security Authorization Process; Mission/Business Process Definition; Insider Threat Program; Information Security Workforce; Testing, Training, and Monitoring; Contacts with Security Groups and Associations; Threat Awareness Program

9.2. Security Management

VA Directive 10-93-142 prohibits local modifications to VistA software.

9.3. General Security

For CAPRI GUI security refer to the following document:

[REDACTED]

This manual includes instructions for setting up CAPRI users, as well as descriptions of all Security Keys used by the CAPRI GUI application.

For AMIE file and VistA-specific security see the CAPRI GUI User Manual at:

[REDACTED]

9.3.1. Remote Systems

The AMIE software does not transmit data to any remote systems. For CAPRI interactions with remote systems, refer to the Systems Architecture diagram in Section 3.1.1.

9.3.2. Contingency Planning

Your facility should have a local contingency plan in the event of application problems in a live environment. It should identify the procedure for maintaining functionality provided by the AMIE software as well as the CAPRI GUI application, in the event of system outage.

9.3.3. Interfacing

There are no special interfacing requirements for the AMIE or the CAPRI software.

9.3.4. Electronic Signatures

The CAPRI GUI application uses electronic signatures. Use the following link to locate the CAPRI GUI User Manual: [REDACTED]

9.3.5. Security Keys

Take the following steps to get information about the security keys used with the AMIE software.

1. VA FileMan Menu
2. Print File Entries Option
3. Output from what File: SECURITY KEY
4. Sort by: Name
5. Start with name: DVBA to DVBC
6. Within name, sort by: <RET>
7. First print field: Name
8. Then print field: Description

***Note:** Some keys do not affect the menu operation. This is due to some options having several different functions which are limited in scope by the key. This limitation is done internally by the program being used.*

9.4. Privacy

- 9.4.1. Privacy is maintained through system access restrictions based on [VA DIRECTIVE 6500 InformationSecurity](#)

9.5. File Security

Find File Security information in the: [CAPRI System Administration and Technical Guide](#)

10. Approval Signatures

The signatures below are an acknowledgement that the signatory understands the purpose and content of this document.

X

[REDACTED]
IPT Chair

From: [REDACTED]
Sent: Thursday, September 12, 2013 6:42 PM
To: [REDACTED]
Subject: RE: **Your Approval of the System Design Document is requested**Patch 187 CAPRI MS1

I concur

X

[REDACTED]
Business Sponsor

From: [REDACTED]
Sent: Tuesday, September 17, 2013 9:41 AM
To: [REDACTED]
Subject: RE: ** [REDACTED] please read** Approvals Needed

Classification: [Not VA Sensitive// Not VA Record](#)

I approve the documents listed below.

RSD, ACP, SDD and PMP.

[REDACTED]
[REDACTED]
[REDACTED]
[REDACTED]
[REDACTED]
[REDACTED]

X

HP Project Manager

From: [REDACTED]
Sent: Thursday, September 12, 2013 6:22 PM
To: [REDACTED]
Subject: RE: **Your Approval of the System Design Document is requested** Patch 187 CAPRI MS1

I concur

[REDACTED]
[REDACTED]
[REDACTED]

X

[REDACTED]
[REDACTED]



AERB APPROVAL

X

Service Delivery and Engineering

From: [REDACTED]
Sent: Friday, September 13, 2013 11:18 AM
To: [REDACTED]
Subject: RE: **Your Approval of the System Design Document is requested** Patch 187 CAPRI MS1

I Concur

Thank You for the nice job and effort with these sections 1 through 3 for the upcoming MS1 review, I found the information informative and helpful.